Knowledge Components and Methods for Policy Propagation in Data Flows

Thesis

Knowledge Components and Methods for Policy Propagation in Data Flows

ENRICO DAGA

Knowledge Media Institute

Faculty of Science, Technology, Engineering and Mathematics

The Open University

This dissertation is submitted for the degree of

Doctor of Philosophy

October 2017

Abstract

Data-oriented systems and applications are at the centre of current developments of the World Wide Web (WWW). On the Web of Data (WoD), information sources can be accessed and processed for many purposes. Users need to be aware of any licences or terms of use, which are associated with the data sources they want to use. Conversely, publishers need support in assigning the appropriate policies alongside the data they distribute.

In this work, we tackle the problem of policy propagation in *data flows* - an expression that refers to the way data is consumed, manipulated and produced within processes. We pose the question of what kind of components are required, and how they can be acquired, managed, and deployed, to support users on deciding what policies propagate to the output of a data-intensive system from the ones associated with its input. We observe three scenarios: applications of the Semantic Web, workflow reuse in Open Science, and the exploitation of urban data in City Data Hubs. Starting from the analysis of Semantic Web applications, we propose a data-centric approach to semantically describe processes as data flows: the Datanode ontology, which comprises a hierarchy of the possible relations between data objects. By means of Policy Propagation Rules, it is possible to link data flow steps and policies derivable from semantic descriptions of data licences. We show how these components can be designed, how they can be effectively managed, and how to reason efficiently with them. In a second phase, the developed components are verified using a Smart City Data Hub as a case study, where we developed an end-to-end solution for policy propagation. Finally, we evaluate our approach and report on a user study aimed at assessing both the quality and the value of the proposed solution.

To Filippo and Pietro
(Dad wrote a book!)

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

Enrico Daga (October, 2017)

Acknowledgements

I can fairly say that this has been a long journey since when I wrote my first line of code 13 years ago at the age of 27. The path leading me here has been full of serendipitous events, among which the most important was attending a language course at the Russian Institute of Culture in Rome. Another one is related to a newspaper advert of a training course for "Expert Web Developers", kindly cut out by a friend of mine for me, as I was unemployed at that time.

I will skip many others and only summon a conversation between who is writing and a young but already distinguished researcher during a meeting of the EU-funded NeOn project:
- *"So, are you doing a PhD in Semantic Web?"*
- *"No. Anyway, I could not do one in Computer Science, I have a degree in Humanities. I am just a developer who likes Semantic Web."*
- *"Well, I am pretty sure you could do one with us at The Open University!"*

First and foremost, I offer my sincerest gratitude to my supervisors, Professors Mathieu d'Aquin, Enrico Motta, and Aldo Gangemi, for being my guide during these PhD years and showing me the art of research whilst allowing me the freedom to work in my own way. I am very grateful to the reviewers Oscar Corcho (Universidad Politécnica de Madrid) and David Robertson (University of Edinburgh) for the constructive feedback and for having made my PhD viva a memorable experience, and to Trevor Collins who had a key role in reducing my anxiety before and during that day.

Many excellent colleagues at the Knowledge Media Institute (KMI) at The Open University supported me with through their valuable feedback. The superb Miriam Fernandez read the whole draft returning precious feedback and challenging questions which prepared me for the review panel. Paul Mulholland and Francesco Osborne provided valuable feedback and helped me with solid suggestions on how to improve the work in Chapter 6.

I have been rewarded by a friendly and cheerful group of colleagues and friends that supported me during these years: Alessandro Adamou, Luca Panziera, Ilaria Tiddi, Emanuele Bastianelli, Andrea Mannocci, Carlos Pedrinaci, Matteo Cancellieri, Alessandro Taffetani, Patrizia Paci, Jaisha

Bruce, Lucas Anastasiou, Pinelopi Troullinou, Lara Piccolo, Martin Holsta, Giorgio Basile, Simon Cutjar, Tina Papathoma, Allan Third, ... and all the people of the Knowledge Media Institute of The Open University. Thank you for making KMI the ideal place for a PhD, and for being so sensitive in supporting the cause of inclusiveness and diversity in these politically difficult times.

A special acknowledgment goes to the Knowledge Media Instruments for being "The Best Band You Ever Heard in Your Life" (Cit.). Follow them on Facebook: `https://www.facebook.com/kmiinstruments/`.

I would never have thought of starting a career in research without having had the opportunity to collaborate with the Semantic Technology Laboratory (STLab) of the Institute of Cognitive Sciences and Technologies (ISTC) of CNR: Aldo Gangemi, Valentina Presutti, Andrea Nuzzolese and their first-class team of researchers and collaborators.

I am forever grateful to Alberto Salvati for having believed in me at a time when I did not. Those years at the Ufficio Reti e Telecomunicazioni and later at the Ufficio Sistemi Informativi of the Italian National Research Council remain unforgettable. The strongest hugs go to Gianluca Troiani and Andrea Pompili, brothers in arms. I can fairly say I learned from the best.

A special thought goes to my family and friends in Italy: Andrea (Gino), Pietro, Alessio, Alessandro (Cuccioletto), Enrico (Abbruzzese), Massi and all the metal-heads of Decima[1]. I want to thank my steady mum for being a constant in my life in many different ways (- *"Mum! I am PhD!";* - *"What does it mean?"*), my brother Marcello (admirable perfectionist), and my sister Francesca and her beautiful family. I am grateful to my father Luigi Daga, whose example guides me, lifts me and inspires me daily.

This work is dedicated to my fantastic kids, Filippo and Pietro, who fill my days with joy and purpose. I could not have done this without my wife's support, Francesca. She is capable of making disappear any fear and doubt, by being there when it truly matters.

My Russian is still very basic, though.

[1] `https://goo.gl/maps/NUFGQbjc9jk`.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Data-oriented systems and applications are at the centre of current developments of the World Wide Web (WWW). Citizens, institutions and private organisations publish and consume a large variety of information on the Web. Emerging enterprises focus their business model on providing value from data collection, integration, processing, and redistribution. Nowadays, the extraction, publication, and reuse of data on the Web is an established practice, and a large number of APIs provide access to JSON documents, data tables, or Linked Data for a variety of use cases, spanning from content and media linkage [Kobilarov et al. (2009)] to science and education [Kawase et al. (2013)]. This increase in the ease of data exchange implies that a huge number of information artefacts are exchanged, sometimes propagating with unexpected speed to several different media. These different media have a single thing in common: they benefit from *Web systems*, technologies that are deployed on the World Wide Web (WWW) and that act as mediators between data producers and consumers. The WWW persistently links these systems, so that exchange of information can happen without interruptions and at a very low cost.

The heterogeneity of data artefacts and of manipulation processes on the Web calls for a deep understanding of the way those processes affect the interests associated with such data. Research on data integration considers the problem of heterogeneity as a matter of syntactic (and semantic) mappings between different data sources. However, a large part of the challenges related to integrating heterogeneous data sources on the Web are fundamentally non technical, and can only be understood as we observe the *context* surrounding the way datasets are published and consumed. The contextual information associated with a digital object (datasets included) is called *metadata* - the data about the data. In this work we focus on a specific type of metadata: the *policies* derivable from the *licence* or *terms of use* associated with the data. In particular, we tackle the problem of policy propagation in *data flows* - an expression that refers to the way data is consumed, manipulated

and produced within *processes*.

Let's take few scenarios. A civic officer is compiling a report about the quality of green areas as it is perceived by the inhabitants of the council. For this activity, a developer is engaged with the task of collecting geolocated tags from social media platforms, combining it with statistics about local crimes taken from the police, and represent this information in a map (for example, reusing Open Street Maps). *How to decide what terms of use need to be assigned to the data included in the report?*

Similarly, a scientist elaborates data produced by a private company, which forbids to use the data for commercial purposes. However, the experiment performed will produce new data, on the basis of the ones regulated by those terms of use. *Will the output of the scientist's research have the same restrictions?* Also, those terms of use allow the scientist to perform the experiment but not to share the data with third parties. *How to decide whether the data produced by the experiment itself can be published openly on the Web?*

Finally, suppose a developer is responsible for building a system to exploit Web data to recommend places and events associated with user's location and interests. Such system will reuse data from a variety of Web APIs, but also gather media contents, calendars of events from the websites of local cinemas, theatres, museums and sports venues, and offer an interpreted selection of these data within its own Web API. *What will be the terms of use to publish alongside the API?*

On the Web of (open) data, users interact with a large variety of information. Hence, they need to be aware of any usage constraints attached to data sources they want to use and publish the appropriate policies alongside the data they distribute. In this scenario, assessing what policies propagate from the licenses associated with the data sources to the output of a given data-intensive process is an important problem.

1.1 Motivation

The Web of Data (WoD) is by definition decentralised, and brings new technical challenges in terms of scalability and interoperability [Auer and Hellmann (2012)]. Users [Krumm et al. (2008)], services [Alonso et al. (2004); Studer et al. (2007)] and agents [Jiang et al. (2010)] contribute to the development of the WoD by publishing, linking [Qiu et al. (2004); Rodrigues et al. (2011); Volz et al. (2009)], sharing, elaborating and consuming data. Moreover, it is expected that new types of agents will be active on the Web, namely IoT sensors [Guinard and Trifa (2009); Guinard et al. (2011)] and autonomous robots [Waibel et al. (2011)]. It can be argued that a large quantity of the content published on the Web is the result of the elaboration of content consumed from the Web

itself.

The multiplicity and diversity of owners, interests, licenses and terms of use that can be part of policies metadata opens new challenges. The first challenge concerns the way policy metadata can be represented on the Web. The W3C ODRL Community Group curates the Open Digital Rights Expression Language (ODRL) [Iannella (2001, 2002); Iannella et al. (2012)], a standard for the representation of policies on the WWW, recently published as recommendation [Steidl et al. (2018)]. ODRL can be used to make statements about *permissions*, *obligations* and *duties* to support negotiation and control of access and usage of resources on the Web. Research on policy representation and reasoning provide the theoretical foundations and practical techniques to reason with policies, making it possible to check the compatibility of licences and the consistency of requirements and actual usage, when represented with the appropriate formalism.

Policy representation and reasoning have received considerable scrutiny in the past few years. While the problem of licence compatibility has been studied and technologies for reasoning on licenses and policies compatibility have been developed, there is a need to understand the way policies propagate through the processes established in systems relying on distributed datasets. Performing this assessment on a case-by-case basis can be overwhelming. Data publishers would avoid exposing their data on the Web to prevent potential misuses. Data consumers might decide to simply not use any Web data at all, feeling unsure about how to handle the licensing issue. This problem is of fundamental importance in the Web of Data, and it can be observed in three scenarios: applications of the Semantic Web, workflow reuse in Open Science, and the exploitation of urban data derived from complex processing pipelines in City Data Hubs. In the next sections we will analyse why the interaction of datasets, policies and processes is a foundational aspect of the three scenarios, and will detail the key motivations underlying our research on policy propagation in data flows.

1.1.1 Semantic Web applications

The Semantic Web research programme was initiated by the W3C at the beginning of the new century with the objective of making the Web a global shared *intelligent* system. In the Semantic Web vision, agents explore, discover, select and access resources autonomously, and create and exchange new ones transparently by exploiting the metadata associated with data and services [McIlraith et al. (2001); Pedrinaci et al. (2011)]. Over time, this programme has produced resources, methods and technologies with the purpose of enabling Web Systems to seamlessly interchange information and autonomously reason upon them. The Resource Description Framework (RDF) is the underlying meta-model that establishes a graph based structure to represent knowledge at potentially any level

of granularity. A set of languages have been developed to add *semantics* to RDF data, primarly the RDF Schema (RDFS) specification. The Web Ontology Language (OWL) was built with the purpose of enabling automatic reasoning over Web data, in the tradition of *description logics* [Antoniou and Van Harmelen (2004); W3C OWL Working Group (2012)]. Research on knowledge representation and reasoning contributed largely with methodologies and tools to support the development of ontologies for the Web [Blomqvist et al. (2010); Daga et al. (2007); Staab and Studer (2013)]. The primary objective of the Linked Data paradigm is the *population* of the Semantic Web with more and more data, by exploiting the technologies and ontologies developed so far [Heath and Bizer (2011)].

Semantic Web applications consume Linked Data from a large variety of machine readable resources, perform aggregations and other complex elaborations, and often republish the result as new data on the Web. However, although these technologies reduce the cost of *syntactic* data integration by assuming that the publishing system and the consuming one both share the same underlying models (RDF and shared vocabularies), this activity is far from simple. The Linked Data graph allows indeed users to perform two major tasks in knowledge integration: (a) discovery of relevant data sources, and (b) early analysis of the data, meaning that the adoption of shared vocabularies allows users to easily assess to what extent a data source can be of use in a specific data relying task. However, the activities of selecting and integrating data, as well as carrying out task oriented reasoning need to be defined and performed by the systems that want to exploit the Web of Data. From this perspective, Semantic Web systems can be compared to *Expert Systems* studied at the end of the last century [d'Aquin and Motta (2016)], with the fundamental difference that the former use data which are distributed on the Web. Therefore, while the life cycle of data within Expert Systems can be compared to the way data is managed in the closed environment of organisations and enterprises, the life cycle of data within Semantic Web systems cannot. In the former, data is under the full control of a company, which can set up policies and methods to control access and use of such data. In the latter, the data is exposed for access and reuse by third parties, and the ownership and control over the data and its usage is distributed to different actors and cannot be centrally monitored. Therefore, the Semantic Web community has developed methods and specifications to enrich data with additional information on the way it can be exploited and reused. Of crucial importance are standards like PROV and the related ontology PROV-O [Lebo et al. (2013); Moreau and Groth (2013)], meant to describe the *provenance* of a dataset in terms of involved agents and actions performed [Moreau (2010)]. Datasets can then be linked to this type of metadata in order to inform the consumer about the activity underlining its production. However, the role of provenance information is to describe the nature of the actions performed and by whom

they have been performed, not how they relate to the policies associated to each one of the data items involved. Indeed Semantic Web Systems need to face an open challenge in the way their processes interact with the licences and terms of use of each data source involved. Therefore, it is of primary importance to integrate provenance information with additional metadata to support the developers of Semantic Web systems on the assessment of the way their tasks affect the permissions, prohibitions or constraints associated with the input data and what policies are then applied to the output. If we don't address the licensing issue, on the one hand developers will not be able to assess how policies affect their data, and thus they will not be able to exploit this information effectively; on the other hand data publishers will not be supported in protecting their own interests, and this could prevent them for publishing valuable information, limiting the scope of the Web of Data dramatically, as well as the applicability of Semantic Web technologies. Assessing the policies associated with the data produced by processes exploiting resources on the Web is an important and difficult problem, which can only be solved by studying the way they *propagate*.

1.1.2 Open Science

Data drives research in numerous fields, and while this is obviously due to the fast advancement in technology and infrastructure of the past years, it also has to be credited to the development of a policy regime that supports the open availability and use of research data for scientific progress, promoted by public institutions in various countries (U.S. [Peled (2011)] and Europe [Jasanoff (2011)]). Precursors of these developments were the promotion of the culture of *public domain* developed at the end of the last century [Litman (1990)], the emergence of *open access* as a new paradigm of knowledge sharing developed on the World Wide Web [Craig et al. (2007)], and more recently the call for transparency and reproducibility of scientific procedures, i.e. *scientific workflows* [Altintas et al. (2004); Deelman et al. (2005)]. Scientific results and procedures are today perceived as public goods, and references to the 'information commons' abound in the literature of the past ten to fifteen years [Beagle et al. (2006)]. Open access and reuse is widely recognised to be fundamental for scientific progress.

However, the success of realising the full potential of open science will be the resultant of a variety of different and interrelated social, legal and technical transformations. Indeed, some of the most difficult aspects among these are non-technological in nature. The supposedly *soft elements*, the social and institutional elements of these transformations, are fundamental complements to technical advancements, and need to be as much a subject of research as the technical ones. Among the desirable progresses in the governance of this phenomenon is the development of contractual provisions for licensing data products and services to or from the private sector, with the aim to allow

the research outcomes to have an impact on the economic development of society but at the same time protect the intellectual ownership [David (2004)]. It is arguable that excessively rigid efforts to maintain science data free of private control will necessarily end by yielding less data to the public domain. Conversely, a contractually reconstructed perspective on the data commons, while less ideal in theory, will make more data published and safely accessible in the long run [Reichman and Uhlir (2003)].

Research in the legal domain faces the issue of identifying the specific requirements of the scientific community and identify the assets and policies that should be considered when regulating the research practice, one solution being the development of a specific licence for research assets to protect a researcher's interests and guarantee a positive impact and reuse of research contributions [Stodden (2009)]. There is little doubt that further societal gains in knowledge-creation could be made possible by reducing the social costs of a reliable access to the processing, reproduction and transmission of scientific information. Recently, *scientific data repositories* started providing means to store and publish data related to research articles with the aim to enable persistence and reuse [Amorim et al. (2016); Burton et al. (2015); Candela et al. (2015); Eschenfelder and Johnson (2014)]. Among these are initiatives like Dryad[1] - established by a group of journals adopting a common joint data archiving policy (JDAP) and Zenodo[2], born in the context of the EU FP7 project OpenAIREplus [Manghi et al. (2012)]. The concept of *scientific data* is defined as "entities used as evidence of phenomena for the purpose of research or scholarship" [Borgman (2015)]. This definition applies to a large variety of *data artefacts*, spanning from data tables to relational databases, textual documents or even binary data like photographs, charts etc. The characterisation of the terms of use of these data artefacts (licensing) is fundamental to enable an appropriate and informed reuse by third parties. These terms vary depending on each case, and can include the protection of the interests of the dataset producer (e.g., the requirement for attribution) or the interests of subjects involved in the study, for example a requirement for *data masking* with the objective of protecting personal data [Eschenfelder and Johnson (2014)]. In fact, there is a strong relation between the capability of expressing and enforcing policies and the willingness of opening the data to the public. The experience of the past years demonstrated the role of control on the use of resources have in fostering the publication of research assets [Pryor (2009)].

A major issue is therefore to make researchers confident that the use of the shared data would protect their fundamental rights and this can only be done by understanding how the data is used (or misused) by third parties. The diversity of research domains and data artefacts opens new challenges on how to provide support to publishers on protecting the associated requirements. The variety of

[1]Dryad Website `http://datadryad.org/`

[2]Zenodo Website `https://zenodo.org/`

issues prone to affect usage policies calls for researching approaches to policy management that are agnostic to the specific domain, while being general enough to cope with this diversity.

Scientific Workflows are one class of artefacts in this heterogeneous landscape [Gil et al. (2007)]. A major objective of the research on scientific workflows is to allow reuse. Methods and tools have been developed in the past years in order to support tasks such as ensuring the reproducibility and reuse of research methods and data [Altintas et al. (2004); Callahan et al. (2006); Deelman et al. (2005); Oinn et al. (2004); Wolstencroft et al. (2013)]. Reusable and adaptive workflows are fundamental to the exploratory nature of science. A pre-existing scientific workflow can be applied to new data sources, producing new data to be shared and redistributed. Collaborative methodologies have been proposed to support the reuse of scientific workflows, for example through iterative verification of workflow executions [Gil et al. (2007)]. The emergence of large *grid* infrastructures including different types of resources and *pipelines* opens new challenges in the way licences relating to the different artefacts and stakeholders (data publishers, data consumers and infrastracture managers) may affect the data output of complex computations [Cieri and DiPersio (2015)]. Within this paradigm, a large quantity of *derived datasets* emerge from the original ones, all prone to be affected by the policy requirements associated with the originals. Establishing the way policies can be defined and associated (and enforced) to derived artefacts is an important goal [Zhang et al. (2010)]. For example, it was argued that measuring the impact of dataset reuse could lead to the establishment of a sort of *transitive credit* that could provide a relevant incentive to data publication [Missier (2016)].

Such *derived datasets* can themselves be exposed for reuse for future research. Future users will be required to explore the chain of dependencies between datasets in order to satisfy the policies associated with a given dataset and benefit from the outcome of their activity in a way that respects the rights of the original data providers. Solving the problem on a case by case basis would be overwhelming even for small chains of dependencies. Providing a degree of automation in the process of understanding how policies propagate from the input of a data-relying process to its output within workflow engines or distributed grid infrastructures for scientific experiments is therefore essential to the success of *workflow reuse* in research.

1.1.3 City Data Hubs

The city of the future will be built on substructures of information and communications technologies (ICT) [Kitchin (2014); Naphade et al. (2011); Townsend (2013); Zygiaris (2013)]. Recently, numerous initiatives investigate the vision of a *Smart City*, where cutting-edge technologies are applied to a number of sectors including government services, transport and traffic management,

water, health care, energy, urban agriculture, waste, and resource management. In this vision, traditional networks and services are made more efficient by a combination of technical and non technical solutions, in order to cope with the changes in scale that are envisaged for the city of the future. Beneficiaries of these innovations are the city inhabitants, of course, but also businesses and local governments, in a communication loop that aims to meet demand and supply of services and resources in continuous interaction. The aims of ICT systems deployed in the city will be a mix of observation (sensors, feedback) [Kloeckl et al. (2011); Perera et al. (2014)], reasoning (integration, analysis, decision) [Pan et al. (2013)] and actuation (cyber-physical systems) [Gurgen et al. (2013); Lee et al. (2014)]. Information will flow from sources to targets, building new networks connecting humans and physical devices. These networks are very often overlapping with the Web, the primary medium to reach end users, being citizens, decision makers or infrastructure managers. City Data Hubs are emerging on the WWW as centralised nodes to control and monitor the flows of information between the variety of systems deployed in the territory [Lea and Blackstock (2014)]. The characteristics of these types of systems are physical distribution and co-participation of many different stakeholders. On the one hand, the interests involved in the regulation of the captured data include the rights of the owners of the sensing infrastructures (e.g. sensor networks), who want to control the potential value of the sensed data, and the observed agents (organisations, citizens), whose concerns need to be incorporated in a realistic data governance strategy. On the other hand, the subjects involved in the data processing and consumption need to be aware of these constraints, in order to consider them as requirements for the design of their systems. Current research is targeted at understanding how to govern the life cycle of data in City Data Hubs [d'Aquin et al. (2015a)]. The diversity of data sources, owners and licences associated with the data brings the problem of data exploitability, defined as *the assessment of the policies associated with the data resulting from the computation of diverse datasets implemented within a City Data Hub* [Daga et al. (2016a)]. The issue relates to providing the right level of information regarding the rights and policies that apply to the delivered *derived data*.

The development of Intelligent Semantic Web applications, the reuse of Scientific Workflows in Open Science and the exploitation of data resulting from complex data pipelines in City Data Hubs are three scenarios that have in common the challenge we want to address in the present work. Therefore, the general problem is related to what is required to trace the propagation of policies from the licenses associated with the input data sets to the output of a data relying process.

1.2 Research questions

Semantic Web, Open Science and Smart Cities are three domains in which the problem of policy propagation within data flows on the Web is an important one. However, existing solutions only solve parts of the problem. On the one hand, research on policy representation focuses on the way policy statements can be represented, defining vocabularies of actions to be used in licence descriptions, reasoning on the consistency between licences or with their combination. On the other hand, research on process description is in large part centered on workflow actions reuse, combination and efficient execution, although recently an interest emerged on the way workflow semantics can be elicited and exploited in tasks such as classification and preservation.

Instead, the goal of our work is to identify the components required to trace the propagation of policies from the licences associated with the input data sets to the output of a data relying process. The objective can be articulated in a set of research questions, which cover the various aspects of reasoning on the propagation of policies:

Research Question 1 *What are the knowledge components required to reason on policy propagation?*

Policy propagation is a problem of reuse of licenced data by multiple actors within distributed systems on the Web. In this research, the Web is not only a logical space of interlinked objects but more importantly a material space of assets (e.g. encoded models, software) and human practices (e.g. content and data publishing, resources' access or licencing). Under this light, the initial objective is to identify the elements of this space that are relevant to our solution. Fundamentally, the solution we seek has to be found at design level in order to include all the elements involved, being them a software program, obviously, but also a characterisation of the information this program requires to operate correctly. We observe the policy propagation problem from the perspective of knowledge representation. Therefore, we start by identifying the elements of the solution as semantically characterised information about the assets, schemas, and processes involved. We use the expression *knowledge components* to refer to these semantic elements of our solution.

Research Question 2 *How should we represent the relation between processes and policies in order to practically support the task of reasoning about policy propagation?*

The fundamental problem is to decide when a certain policy in the input ends up to be relevant to the output of a process. Research on policy reasoning studied the problem of actions violating policies extensively. Similarly, a long tradition of research in process modeling focused on how to represent complex sequences of actions (workflows) by the means of basic fundamental components,

for example to enable users to reuse the elements of a process and combine them with others to build new processes. However, if the objective of our research is to tackle policy propagation, both policies and processes should be represented consistently and such representation should focus on the way process actions affect the propagation. This is a matter of eliciting the *semantics* of the operations performed. In our work, we assume the Open Digital Rights Language (ODRL) to be the reference formalism for the representation of policies[3]. However, independently from the formalism involved on the representation of policies and processes, it is the *interaction* of process and policies that needs to practically support the reasoning process.

Research Question 3 *How can we assist users in setting up the necessary components to enable automated reasoning on policy propagation?*

Ultimately, the objective of our research is to support users that are required to make decisions about policy propagation. With this research question, we consider the problem of how to acquire these knowledge components and how to manage them. Therefore, we want to ensure that the solution proposed is bound to an adequate methodological support. More importantly, we intend to study how our solution for policy propagation could be integrated in a realistic scenario. It is worth noting that different roles are involved in the development and usage of the knowledge components we presume to be necessary for the task, spanning from the publisher to the consumer of the data and to intermediary agents that can play a part in it. Therefore, this question need to be read into the perspective of the general life-cycle of data in distributed systems.

1.3 Approach

We place our work in the tradition of Knowledge Engineering, with particular relation to bottom-up approaches to knowledge representation and reasoning. Within this perspective, policy propagation in data flows can be seen as a knowledge representation issue, where metadata about datasets and processes play a central role. In order to study the problem, we observed the Web of Data from the point of view of metadata acquisition and management, and therefore derived the design requirements relevant to the implementation of a metadata management method that includes automated policy propagation. The knowledge components required to perform policy propagation are therefore: (a) information about the data sources, ownership and licence; (b) information about the process manipulating the data sources; (c) information about how process steps affect policies at a granular level. Licence *metadata* can be represented by the means of state of the art

[3]The present work is not meant to give a contribution in this area.

techniques, including the DCAT vocabulary and particularly the Open Digital Rights Language (ODRL). In our work we rely on the RDF Licenses Database [Rodríguez-Doncel et al. (2014)], including a catalogue of licences represented in ODRL. Each licence is represented as a collections of permissions, prohibitions, and duties that the licensor grants to the licencee. In this work we refer to each of these atomic elements as a *policy*. Example of policies are: the permission to modify, the prohibition to distribute, or the duty to attribute. Although there is extensive research on process knowledge representation, it is still an open problem to represent the relation between process *actions* and policies of input data. In other words, how to represent the fact that a certain action over a licensed data item will produce new data whose use may affect (or may be affected by) a policy specified in the original licence. In our work we introduce a data-centric representation language to describe processes as networks of data objects: the Datanode ontology. Datanode is made of a hierarchy of OWL object properties that qualify the possible kinds of relations between data artefacts involved in a complex system. By relying on ODRL licences to represent data policies, and on the Datanode ontology to represent process actions, we establish *Policy Propagation Rules (PPR)*. A PPR associates a Datanode relation to a policy, so that in case a source data object includes that policy, it will be propagated to the target data object with which it is related. This technique allows us to reason on policy propagation. However, in order to apply it there are a number of issues to be solved, spanning from the management of a large rule base (number of possible actions times number of possible policies), to the performance issues that such reasoning might bring. We propose a methodology to compress PPRs and experiment with different reasoning approaches. In addition, such techniques need to be offered to users who typically would face the problem of assessing the policies applicable to the output of a process from the ones associated with its input. Therefore, these components need to be collocated within a general data life cycle, supporting users on their acquisition and management. We propose a holistic approach to Data Cataloguing, where information about policies and processes are included in the metadata management activity, and tools support users in the acquisition and management of policy and process knowledge. In this way, we decomposed the major research questions into sub problems, tackled each one of them individually and then re-composed the results in a unified solution, an *end-to-end* approach to policy propagation.

While the status of theories around policy formalisation and reasoning are a significant part of our background literature, the connection between those and the practice is still not completely defined by the state of the art. This statement is motivated by the context of the evolving Web of Data landscape and how it is reflected in different ways in the World Wide Web. Indeed, we chose to see the Web of Data as not only the set of openly published datasets, but we broadened its definition

to also include the data and processes that are in the grey area of being partly or completely closed, but nevertheless affect the way information is published and consumed on the Web. The observed practices are on the one hand the way data are catalogued (and licensed) and the way they are published and consumed, and on the other hand the way data are manipulated by processor systems (for example, scientific Semantic Web applications). The gap observed justified the approaches used to tackle each problem and answer the research questions related to how processes interact with policies and how all this can be part of the general data life cycle.

This approach leaves some theoretical aspects outside the scope of the work. For example, we do not demonstrate the completeness of the models we propose, while we present them as open to evolution, an aspect that is an integral part of the proposal. For instance, because we cannot enumerate all possible policies and activities, our target was to offer a *solution as method*, where the concrete models are prototypical artefacts that are instantiated by the methods proposed.

1.4 Hypotheses

We stress the idea that the problem of policy propagation is primarily at the *knowledge level* [Newell (1982)], and therefore it can be tackled using a knowledge engineering approach. In particular, we reduce it to the understanding of its *knowledge components*, and this leads to the hypotheses of our research:

Hypothesis 1 *It is possible to design a system capable of propagating policies in data flows by following a Knowledge Engineering approach..*

We hypothesise that the solution can be defined completely at the knowledge level with no need for innovative symbol-level elements (i.e. low-level computational optimisations). Our approach identifies the licence (as collections of policies) and the process as basic building blocks of the solution. But how to express the relation between these elements? We conjecture that the relation between processes and policies can be expressed by focusing on the data items involved in the system, and characterise the dependencies that might occur between them:

Hypothesis 2 *An abstract representation of a data manipulation process is sufficient for reasoning on policy propagation. Such abstraction characterises the actions performed as relations between objects in a RDF graph.*

In such a graph, nodes are objects produced or consumed by activities and arcs represent both the type of activity (e.g. copy, split, refactor, etc...) and the direction of the manipulation process (from a input node to an output). According to a graph based representation of a process, the goal of the

reasoning pertains the production of new statements about the policies associated with nodes other then the one representing the input data. Crucially, we need to identify the reasoning properties required to produce such statements. This consideration leads us to the next hypothesis of our reasearch.

Hypothesis 3 *The required reasoning has well-known computational properties such as transitive rules.*

We make the hypothesis that reasoning on policy propagation is feasible with mechanisms that are well-studied and deployable in realistic scenarios without the need of novel logic frameworks or computing models. For example, transitivity is a basilar reasoning property that we could exploit for propagating policies. A rule is a statement of the form *if ... then ...*. A transitive rule is any statement of the form *If a R b and b R c, then a R c*. In this formula, *R* is a particular relation (e.g., "... is the same as..."). Executing rules generates new statements in the knowledge base. For example, if a file *A* is copied into a file *B* and a file *B* is copied into a file *C*, then file *C* is also a copy of file *A*. Such rule can be executed recursively until no new assertions of the type *x R y* are generated. If some data cannot be distributed and the data is copied to an external device and from this device to others, then any of these copies should not be distributed. However, what is important here is that whatever is the reasoning mechanism we propose, it will have to be accessible to anybody with basic computing resources.

Under these hypotheses we design the knowledge components and the associated method for handling policy propagation. However, for this method to be applicable, it must be part of the workflow that developers and data managers establish when dealing with distributed data-relying tasks: publishing, consuming and cataloguing. Therefore, we dedicate particular effort on identifying critical tasks that are crucial for the success of these activities, namely licence selection and process model annotation. Again, we approach the problem by relying on the semantics of process and policy models:

Hypothesis 4 *A semantic model of processes and policies, while being necessary to reason about policy propagation, can also support the acquisition of policy and process knowledge.*

We believe that the time and effort required for assessing what policies propagate from the input to the output of a process can be substantially reduced with the support of the contributions presented in this thesis. However, by doing this we make a set of assumptions, to which we will refer during our work.

Primarily, we consider processes as an ordered sets of actions moving data from one or more inputs to one (or more) outputs, assuming that the output is not sent back as input. In other words, our graphs are meant to represent processes with distinct input and output:

Assumption 1 *A data process transforms data from one or more inputs to one (or more) outputs. Input and outputs are different data artefacts.*

Finally, the designed solution relies on the assumption that there is an environment in which data and processes are controlled and managed:

Assumption 2 *The tools for automated reasoning on policy propagation can be included in the data governance activities, from acquisition to cataloguing and delivery, therefore reducing the cost of assessing the policies applicable to the output of a data-relying process.*

1.5 Research Methodology

The methodology behind the research presented here is akin to Design Science (DS). As research methodology, DS is oriented towards supporting research in the creation of successful *artefacts*. Placed in our scenario, the artefact is a system supporting the reasoning upon the propagation of policies in data flows. The DS process comprises six steps: (1) problem identification and motivation, (2) definition of the objectives for a solution, (3) design and development, (4) demonstration, (5) evaluation, and (6) communication [Peffers et al. (2007)]. The thesis is structured to reflect this process.

The main problem has been decomposed into sub problems, and each one of them has been tackled and evaluated separately. Finally, a global evaluation has been conducted, in order to compose the single contributions into a unified solution for policy propagation. By doing this we followed a constructive approach, trying to answer the research questions by developing tools and methods suitable to solve the problem as configured in concrete scenarios. In our methodology, the *solution design* activity is grounded on the observations of realistic scenarios within the World Wide Web (the Web of Data), particularly in the context of the Semantic Web, Science and Smart Cities.

The Web of Data is an information space that can be observed and studied *in the field*, elements of this digital ecosystem being the artefacts involved as well as the associated practices. These are reflected in the variety of artefacts (systems, licenses, processes and so on) that are part of the observable data landscape within the Web. Therefore, the problem of policy propagation has been observed in the light of existing practices, instead of composing an isolated theoretical issue. We hope that the solutions emerging from the work would improve not only our understanding of the problem, but also reflect the extent to which it can be practically tackled. This last statement is crucial to our research, as we assume the resources published on the Web (the observable part of the existing systems and data) to be representative of the kind of data and processes developers work with. We take this assumption as an epistemological foundation of our research.

1.6 Contribution

Our hypotheses reflect the idea that to reason on policy propagation it is necessary and sufficient to have a representation of the process that expresses the semantics of the activity and put it in relation with the policies associated with the data involved. Semantic Web applications are by definition systems that make extensive use of Web resources. They also have the useful property of being described in scientific papers and technical reports. In fact, papers and reports include a description of the system's functionalities, differently from user documentation, whose scope is limited to how to interact with the system. We looked at Semantic Web applications primarily focusing on the way they manipulate data. We developed the Datanode ontology, a hierarchy of the possible relations between data objects. The Datanode ontology aims to represent a system in a data centric way, as network of data objects. Relying on ODRL as the formalism to express knowledge about policies, we describe how process knowledge and policy knowledge can interact to perform reasoning on policy propagation. We introduce the notion of Policy Propagation Rule (PPR) and describe the production and management of a PPR database. Crucial to this activity is a method to make the management of the rules more efficient as well as validating Datanode as a suitable ontology for the task. Furthermore, as a practical validation, we study the impact of the proposed method on the efficiency of possible PPR reasoners.

After that, we design a methodology for the development of an end-to-end solution to the problem, following the journey of the data from the producer to the consumer, therefore integrating the above processes in the more general one of data governance. We propose a holistic approach to Data Cataloguing, covering the required management tasks for acquiring and maintaining the knowledge components required for policy propagation, and develop a set of tools as exemplary technical solutions to each one of the problems discussed so far. This includes supporting the user on selecting the right licence for the published data by exploiting the semantic representation of the policies declared on each licence. Process knowledge need to be expressed in the catalogue by means of the Datanode ontology, and applying this model to real use cases requires a specific effort. For this reason, we propose a method for supporting users in annotating process knowledge as data-centric graphs, using the scenario of Scientific Workflows. Our method does not require a pre-existing knowledge base of annotated processes. In other words, we bet on the fact that policy propagation can become an integral part of the data governance activity. For this reason, while we evaluate the single components individually, we also dedicate substantial effort on measuring to what extent our solution to the problem of policy propagation is consistent, in other words whether the propagated policies appear meaningful to real users.

The contributions of the present thesis can be summarised as follows:

- Datanode, an ontology of the possible relations between data objects, to be used to semantically represent data flows of applications.
- A methodology to produce and validate a Knowledge Base of Policy Propagation Rules relying on Datanode and ODRL policies, evaluated in terms of ontological coherency and reasoning practicability.
- A methodology and toolkit for data cataloguers, including:
  - A method and a tool to support users in the selection of a machine processable licence to apply to the published data.
  - A novel approach to produce data-centric descriptions of data manipulation processes, applying Datanode.
  - A methodology for the management of the life cycle of metadata about policies and processes that, together with Policy Propagation Rules, allows users to integrate policy propagation into an end-to-end solution.

The next Chapter is dedicated to the state of the art. After a historical presentation of the Web of Data, in the first part we introduce notions and reference literature on the methods and tools that we applied in the work, namely Knowledge Engineering, Formal Concept Analysis, and Semantic Web Technologies. The second part covers background research on process modelling, policies modelling and metadata management, the context within which we place our approach and contribution. Chapter 3 describes the genesis of the Datanode ontology, a formalism for data-centric representation of systems (data flows), resulting from the analysis of a set of exemplary Semantic Web applications. Relying on ODRL for the representation of Licenses and Terms and Conditions documents, and on Datanode for the representation of data flows, we establish Policy Propagation Rules as the necessary component required to perform reasoning on policy propagation. In Chapter 4 we describe a methodology to produce and validate a Knowledge Base of Policy Propagation Rules, and we test its applicability to concrete use cases. The proposed solutions are incorporated into a holistic approach to data cataloguing that allows users to implement an end-to-end solution for policy propagation, described in Chapter 5. The contributions listed so far are evaluated within a unified solution in Chapter 6. Finally, in Chapter 7 we derive some conclusions and insights, discuss relevant open issues, and identify the basis for potential future work. Spftware and ontologies developed for this work have been published on an Open Access repository and had Digital Object Identifiers (DOIs) assigned. A summary is provided in Appendix A for reference.

1.7 Publications

The chapters mentioned above are based on the following publications.

Chapter 4

- Enrico Daga, Aldo Gangemi, Enrico Motta. *Reasoning with data flows and policy propagation rules.* In Semantic Web, 1-21 (IOS Press, 2018)
- Enrico Daga, Mathieu d'Aquin, Aldo Gangemi, Enrico Motta. *Propagation of policies in rich data flows.* Proceedings of the 8th International Conference on Knowledge Capture (K-Cap). New York, USA, 2015.

Chapter 5

- Enrico Daga, Mathieu d'Aquin, Enrico Motta, Aldo Gangemi. *A bottom-up approach for licences classification and selection.* Proceedings of the International Workshop on Legal Domain And Semantic Web Applications (LeDA-SWAn) held during the 12th Extended Semantic Web Conference (ESWC). Portroz, Slovenia, 2015.
- Enrico Daga, Mathieu d'Aquin, Aldo Gangemi, Enrico Motta. *An incremental learning method to support the annotation of workflows with data-to-data relations.* Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 2016.
- Enrico Daga, Alessandro Adamou, Mathieu d'Aquin, Enrico Motta. *Addressing exploitability of Smart City Data.* IEEE International Smart Cities Conference (ISC2). Trento, Italy, September 2016.

Chapter 6

- Enrico Daga, Mathieu D'Aquin, Enrico Motta. *Propagating Data Policies: a User Study.* Proceedings of the 9th International Conference on Knowledge Capture (K-Cap). Austin, Texas, USA, 2017.

Chapter 2

State of the Art

In this chapter, we cover the background of related research areas with the purpose of equipping the reader with notions required to understand the perspective taken, the problems discussed, and the approaches adopted in the present work. We deal with the problem of policy propagation in the Web of Data adopting a Knowledge Engineering perspective. Figure 2.1 illustrates our perspective in this literature review chapter. We will start by giving a background of the emergent Web of data, which is our reference context, in the next Section 2.1. After that we move on to introducing our research background, starting from a short overview of Knowledge Engineering (Section 2.2), which is at the core of our approach, and two fields from which we inherit most of the methods and the concepts adopted, namely Semantic Web Technologies (Section 2.4) and Formal Concept Analysis (Section 2.3). The second part is dedicated to research areas in which we place our contributions, covering foundations and current research in data cataloguing (Section 2.5), policies modelling (Section 2.6), and process modelling (Section 2.7). We provide details of the state of the art in the above areas and clarify the *gaps* that we address in our work.

2.1 The Web of data

Many human activities are nowadays performed by means of the World Wide Web (WWW), a large, open and distributed *information space* [Wikipedia (2016c)]. The WWW started as an application of the Internet protocol TCP/IP [Braden (1989)] with the objective of publishing texts and connecting them with cross-references (links) [Berners-Lee (1989)]. Initiated at the end of the last century, it gradually evolved into the largest collaborative human construction in history. At the very beginning, information meant textual *documents*. In fact, the success of the Web as infrastructure was due to the capability of making a document *available* from remote locations by

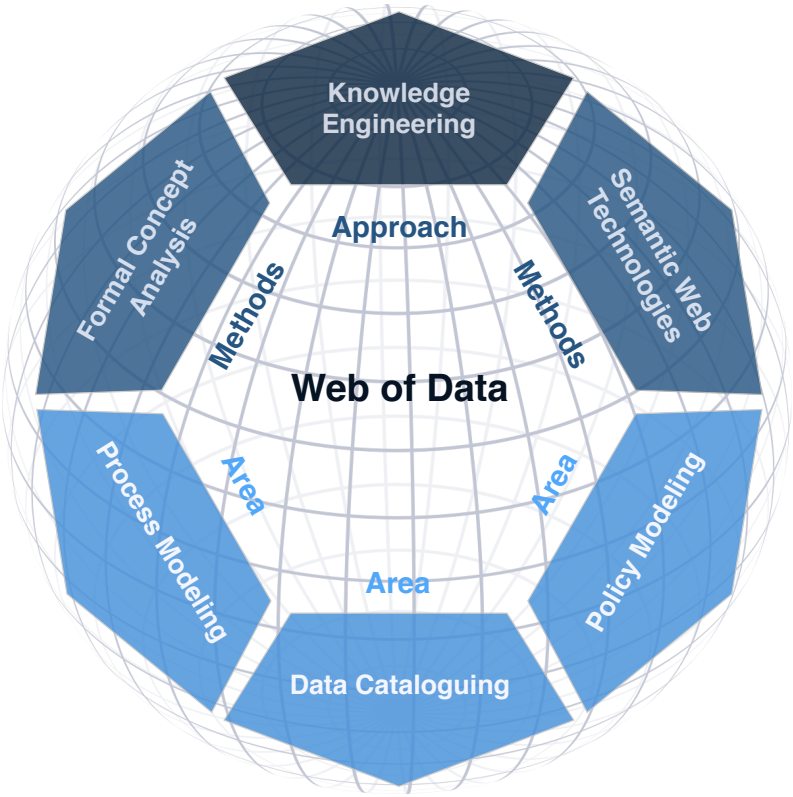


Figure 2.1: Overview of relevant research areas and approaches.

only knowing the corresponding Uniform Resource Locator (URL) [Berners-Lee et al. (1994)], and enabling the discovery of related documents by means of *hyperlinks* implemented thanks to the Hypertext Markup Language (HTML) [Berners-Lee (1993); Pfeiffer et al. (2014)] and the Hypertext Transfer Protocol (HTTP) [Fielding et al. (1999)].

The first major milestone of this evolution was the possibility of users to modify documents collaboratively (the so-called Web 2.0) [Wikipedia (2016b)]. At the technical level, this change is characterised by the advent of Content Management Systems (CMS). A prominent example of this is Media Wiki [Barrett (2008)], the software behind Wikipedia, the global and collaborative encyclopaedia, which is constantly edited by a large number of people organised in *communities* on specific topics of interest [Bruns (2008)]. At this stage, information meant the *content* of documents, which can be changed by multiple editors.

The advent of portable devices permitted more and more people to have constant access to the Web, making it also a platform for social interaction (the so-called Social Web) [Kaplan and Haenlein

(2010); Kietzmann et al. (2011); Lenhart et al. (2010); Shirky (2011)]. The consequence was the Web playing the role of a broadcasting platform, starting a golden age of marketing opportunities. On the one hand, this change affected the way products are commercialised, advertised and sold to customers. On the other hand, a large amount of information about users' everyday life started to become observable, including places visited and events attended as well as images, videos and in general digital objects associated with them. Traces of users' navigation became valuable information to be exploited for advertising, but also for content and experience customisation (user's *profiling*) [Papadopoulos et al. (2012); Tseng and Lin (2006)]. Public bodies recognized the necessity of being *on the Web* and use it to improve communication and interaction with the citizens, and their engagement in public life [Janssen et al. (2008)]. To better exploit the value of this huge amount of digital objects, Content management systems (CMS) were joined by Web Services [Alonso et al. (2004); Erl (2004)]. A Web Service is an Application Programmable Interface (API) [Wikipedia (2016a)], a system exposed on the Web that can be exploited by remote software agents, to *pull* and *push* content automatically. With this approach, websites became no longer targeted to human users only, but to machines as well, and companies that could now build their entire business infrastructure on the Web. In this phase of the Web, information meant *structured data*, consumed and modified in real time by humans and machines seamlessly. The WWW started as an infrastructure for publishing (hyper-)texts, developed in a distributed platform of services.

The constantly increasing amount of services available on the Web is the result of a revolution in the way machines acquire, process and apply information: the so-called Web of Data. This transformation, which is still ongoing, affects a broad range of sectors, for example, Education, Commerce, Science, Mobility, Logistics, Urban development, and so on. The World Wide Web Consortium (W3C) defines a set of standards aimed to harmonise the way data is published and consumed: the Resource Description Framework (RDF) [Lanthaler et al. (2014)] and the Linked Data (LD) paradigm [Bizer et al. (2009)]. Compared to Web APIs, LD is a homogeneous method for data publishing, based on the same approach as the one enabled for the publishing of documents by the early Web. Just like documents are indexed by their location (URL), entities in Linked Data (both things and their properties) can be identified by equivalent Uniform Resource Identifiers (URIs) [Berners-Lee et al. (2005)], so that information can be represented as links. In Linked Data, the city of Rome is `http://dbpedia.org/resource/Rome` (distinguishable from its football team: `http://dbpedia.org/resource/A.S._Roma`) and has the property of *being in Italy* through the link `http://dbpedia.org/ontology/country` to the entity `http://dbpedia.org/resource/Italy`. Large interlinked (open) data sets are becoming

first-class citizens of the data landscape of the World Wide Web (WWW) [Heath and Bizer (2011)], in domains such as education, cultural heritage, academia and research [Gangemi et al. (2011); Haslhofer and Isaac (2011); Zablith et al. (2011)]. In this vision, information *is* the Web of Data, the global, distributed graph of interlinked datasets.

2.2 Knowledge Engineering

Knowledge Engineering (KE) is a computer science discipline that studies methods and tools for the representation and acquisition of knowledge. In general, KE is concerned with technological and representational aspect but also human factors, making it a discipline situated in a middle ground between computer science, cognitive studies and human-computer studies [Aussenac-Gilles and Gandon (2013)]. KE is at the core of the approach we take on analysing the problem of policy propagation and leads to the choice of focusing on the *knowledge components* required and the way they are acquired, developed and managed (see the research questions in Section 1.2). KE developed as a discipline from Allan Newell's 1982 paper "The Knowledge Level" [Newell (1982)], where it was argued that systems can (and should) be designed and modelled independently from their implementation in concrete software artefacts, by distinguishing this representation layer - the so called *symbol level*, from another more abstract one related to *knowledge*. The main characteristic of a *Knowledge System* is to incorporate a large quantity of information, facts (data) or statements (rules), in order to perform knowledge-intensive tasks [Stefik (2014)]. Under this perspective, the general problem is the one of *knowledge transfer*, meaning the activity of incorporating human knowledge into an intelligent system capable of reasoning over a specific family of problems with the accuracy of an expert in the field. Approaches to the development of such systems are targeted to the collection of relevant information and its organisation in a way that is both cognitively relevant and machine processable. Therefore, and in some sense similar to what happened to software development with Software Engineering [Studer et al. (1998)], the field of *Knowledge Engineering* emerged as the attempt to formalise and make reproducible the processes, methods, and tools to design and build Knowledge Systems at scale.

The initial strategies for the development of knowledge systems were essentially based on a plain representation of information in the symbolism of an inference engine, for example, production rules, Prolog, or LISP. These first generation systems (and related approaches), while being successful on incorporating the human's expertise in specific areas, lacked the capacity of being adaptable to different domains. To tackle this problem, further research was centered on *reuse*, and *second generation* methodologies focused on the development of reusable abstracted *components* to be

instantiated in specific domains. In [Clancey (1985)], Clancey made the observation that, while developed using different representation formalisms, many first-generation expert systems could be classified under a common model, called "Heuristic Classification". This led to a consequent shift of the problem from the one of encoding expertise to the one of identifying the *knowledge components* to be used in the development of these systems (compare [Breuker and Wielinga (1987); Bylander and Chandrasekaran (1987); Motta (1997); Steels (1990)]), and by extension in the definition of generic methodologies for the acquisition and design of knowledge systems based on them. These methodologies were based on the hypothesis that the development of knowledge systems was essentially a *model construction* problem, therefore methods and tools for the definition, formalisation and reuse of such models were required [Studer et al. (1998)]. For example, the well known KADS methodology [Breuker and Wielinga (1987)] - later on developed in CommonKADS [Schreiber (2000)] - identified three distinct Knowledge layers, the ones of *Domain* - the capacity of the system to capture the facts relevant to the problem; *Inference* - the ability of the system to reason upon the domain; and *Task* - its ability to solve problems and making decisions, for instance by planning or searching. Under this perspective, the notion of Problem Solving Method (PSM) was introduced with the challenge of identifying the generic inference actions and knowledge roles under which it would have been possible to develop domain independent expert systems. Various PSMs can be found in the literature: *Propose-and-Revise* was targeted to solve problems of parametric design; *Cover-and-Differentiate* for diagnostic tasks, and so forth [McDermott (1988)]. PSMs are based on the *strong interaction problem hypothesis*, stating that the properties of domain knowledge are entirely determined by its use [Bylander and Chandrasekaran (1987)]. Therefore, while PSMs were developed as abstractions to be reused in different - but equivalent in terms of task and goal - domains, the actual models resulting from the instantiation of the PSMs could not be transferred to another system working in the same domain but for a different type of goal. With the emergence of the World Wide Web (WWW), this requirement became more evident, as for the first time knowledge acquisition could be performed at a scale that was impossible before, and elements like interdependencies of systems and procedures became a constituent of the Web of Data. In other words, a new requirement for *interoperability* emerged. As a result, the KE research community - and particularly the research focusing on Knowledge Acquisition - centered its effort on the development of *Ontologies* as domain relevant but transferable knowledge models. The term Ontology was *stolen* from Philosophy, where it identifies the study of the *essence* and *being*. In [Gruber (1991)] a definition of Ontology is made as a *common* model, having, on one hand, the property of being an accurate and shared representation of a domain, and on the other hand, the characteristic of being reusable in various applications (also

compare [Gómez-Pérez and Benjamins (1999)]). Researchers in KE in large part espoused the vision of the Semantic Web [Berners-Lee et al. (1998)], where the WWW as a whole is conceived as a large and distributed knowledge system [Schreiber (2013)]. While the WWW provided the infrastructure for building large-scale distributed systems, KE research contributed by providing the background in knowledge modelling necessary for the development of representation languages like the Web Ontology Language (OWL) or the Simple Knowledge Organization System (SKOS) [Schreiber (2013)]. Research in the Semantic Web faced most of the crucial problems of KE, in particular the Knowledge Acquisition one, and made it evident that knowledge models were not only necessary for designing and operationalise systems, but also that they had inherent value as a means to access, browse and query information at scale [Aussenac-Gilles and Gandon (2013)]. Ontology Engineering in the Semantic Web inherited the research tradition of KE and developed sub-areas like Ontology Evolution, or Alignment, and particularly developed the notion of *reusable components* for instance developing Ontology Design Patterns (ODP) [Gangemi (2005)] and its associated Extreme Design (XD) methodology [Blomqvist et al. (2010); Daga et al. (2010)] for the collaborative development of ontologies also based on knowledge reuse (also compare [Presutti et al. (2012); Suárez-Figueroa et al. (2012)]). In the present work we make extensive use of Semantic Web technologies and specifications Therefore Section 2.4 is dedicated to the basics on RDF, OWL, Linked Data and associated inference capabilities.

An important consequence of the evolution of the World Wide Web (WWW) is related to a number of resources it made available, to be used as background information for knowledge acquisition. While early Knowledge Acquisition (KA) was primarily focused on the elicitation of information from human experts, providing solutions similar to the ones of requirements collection in Software Engineering, for example using Competency Questions (CQ) [Grüninger and Fox (1995); Uschold and Gruninger (1996)], recent approaches focus on the way non ontological resources can be exploited to support ontology design [Villazón-Terrazas (2012)]. The focus on abstraction and reuse did not prevent the development of bottom up approaches to model construction from data [Van Der Vet and Mars (1998)], and semi automatic and supervised techniques to generate ontologies from resources like documents, media objects, folksonomies, Web sites or microblogs (for example compare [Alani et al. (2003); Biemann (2005); Brewster et al. (2002); Fortuna et al. (2006); Maedche and Staab (2000); Navigli et al. (2011)]). This rather difficult task requires to face several problems including terminology extraction [Pazienza et al. (2005)], concept extraction, named entity recognition, relation extraction, including techniques of Knowledge Discovery (KD) [Jicheng et al. (1999); Mladenić et al. (2009)]. In the present work, we employ several techniques from the literature to support the activity of model construction. In the next section, we introduce

Formal Concept Analysis (FCA) [Wille (2005)], a technique widely used for automatic ontology construction from data that we use extensively in our work.

We consider Knowledge Engineering as a research *approach*. In particular, we inherit the notion of *knowledge component* as the basic building block for system design. Before closing this section it is worth spending some words on one aspect of KE methodologies that is particularly present in our work: the iterative nature of model construction methodologies.

KE as a discipline is centered on the development of abstract and generic approaches to system design, therefore substantial effort has been made to study the knowledge elicitation aspect. With a clear focus on making computers and humans interact through *knowledge*, KE recognized the iterative nature of model construction at an early stage [Partridge and Wilks (1987)], for example considering the expertise capture and domain conceptualization as two stages of an iterative process [Motta et al. (1990)]. A comparison between the engineering life-cycle of Software and Knowledge engineering made in 1989 identified a crucial difference between the so-called *waterfall* model of Software Engineering and the more agile and iterative approaches developed in the context of the KE community [Wilson et al. (1989)]. Knowledge construction as iterative process is a notion that is widely adopted by methodologies developed in KE and, more recently, in Ontology Engineering (RUDE [Partridge and Wilks (1987)], CommonKADS [Schreiber (2000)], and Extreme Design (XD) [Daga et al. (2010)], just to mention a few examples). It is worth noting how history proved it the right approach, as recent Software Engineering methodologies have a strong accent on iterative and test-based development methods as well as continuous integration in collaborative projects [Moniruzzaman and Hossain (2013)].

This short background cannot be considered exhaustive, and we refer the reader to [Motta (1997); Stefik (2014); Studer et al. (1998)] for further insights on Knowledge Engineering as research area, and to [Motta and Wiedenbeck (2013)] for a recent historical overview and perspective on the field. The literature on automatic ontology construction is vast, we refer the reader to [Astrakhantsev and Turdakov (2013)] for a recent survey. In our work, we make extensive use of one technique to support a data-oriented ontology construction, namely Formal Concept Analysis.

2.3 Formal Concept Analysis (FCA)

Formal Concept Analysis (FCA) is a conceptual framework for data analysis. Based on lattice theory, from which it derives its mathematical foundations, FCA establishes an abstract model based on binary associations between the entities and their attributes, which permits to derive a browsable classification of clustered *concepts*. Firstly introduced in 1982 by Rudolf Wille [Wille (1982)],

Table 2.1: Example of products and features.

-	Wearable	Powered	Pocketable	Solid	Liquid	Toy	Edible
Hat	x		x	x			
Drone		x		x		x	
Lollypop			x	x		x	x
Ice-Cream				x	x		x
Coffee					x		x
Doll			x	x		x	
Desktop		x		x			
Laptop		x		x			
Watch	x	x	x	x			
Ring	x		x	x			
Shoes	x			x			
Detergent					x		
Whiskey					x		x

concept lattices received immediate adoption in several areas of Computer Science and later on Ontology Engineering as a method to automatically derive conceptualisations from raw data. In what follows we summarise the theoretical foundations of FCA giving insights into the techniques we use, and provide some references about areas of applications of FCA. We also will not go too far into the mathematical details, limiting ourselves to the exposition of terminology and symbols relevant to our work. For an introduction to FCA and its foundations, we refer to [Wille (2005)]. A general overview of the field in the context of information science is [Priss (2006)], including a discussion linking FCA to theories outside strict computer science (for example philosophy and cognitive sciences).

FCA is a knowledge processing technique based on lattice theory. In particular, it exploits the notion of *Galois connection* [Erné et al. (1993)], which refers to a duality that can be observed between two types of items in mutual relation, for example, documents and terms, or products and features. A Galois connection implies that if one makes one of the two sets larger, the second will necessarily be smaller, and vice versa. For example, the more the features we consider, the fewer products share them. Conversely, a single product will include many features, while only some of those will be shared with other products. In Formal Concept Analysis, we refer to these two sets as *objects* and *attributes*. Objects and attributes and the relation between them can be represented as a cross table (also known as incidence matrix), where the row represent objects and the columns their attributes. Table 2.1 shows an example of products and features.

This basic representation is called Formal Context, and is defined as follows:

Definition 2.1 (Formal Context) *Given a set of objects X, a set of attributes Y, a* Formal Context *is defined as a triplet $\langle X, Y, I \rangle$, where $I \subseteq X \times Y$ is a binary relation representing the incidence of*

Table 2.2: Example of *formal concept*. The concept *intension* is the set $\{Wearable, Solids\}$, while the *extension* is $\{Hat, Watch, Ring, Shoes\}$

-	Wearable	Solid	Liquid
Hat	x	x	
Watch	x	x	
Ring	x	x	
Shoes	x	x	
Drone		x	
Coffee			x
Doll		x	
Desktop		x	
Laptop		x	

.

X *and* Y.

Mathematically, a Formal Context is equivalent to a *binary matrix*.

An important feature of Galois connections is that one can assume a *closure* of the relation between subsets of objects and attributes. From this mathematical property, FCA derives the notion of *closed item set*, also known as *Formal Concept*. Following the example in Table 2.1, if one selects the common properties of rings and shoes she will obtain the two attributes Wearable and Solid, as they do not share any other. However, they are not the only solid and wearable products in the list. If then she derives all the objects that are wearable and solid, the set will also include hats and watches. At this point the relation is *closed*, meaning that no other objects will share the two attributes and no other attribute is shared by the resulting set of objects.

Formally, it is possible to derive a concept starting from any object x in the Formal Concept, as follows. Let's consider a *closure operator* $'$ such that $x' = Y$, Y is all the attributes of the object x. The same operator can be re-applied to the obtained attribute set $Y' = X$, returning all the objects having them. The pair $\langle X, Y \rangle$ is a *closed item set*, or *Formal Concept*, where X and Y are the *extent* and *intent* of the concept, respectively. (This process can be followed starting from any attribute y in the same way, such that $\langle y', y'' \rangle$ is also a concept.)

Definition 2.2 (Formal Concept) *A formal concept in* $\langle X, Y, I \rangle$ *is a pair of sets* $\langle A, B \rangle$ *of* $A \subseteq X$ *and* $B \subseteq Y$*, such that* $A' = B$ *and* $B' = A$*.*

In other words, A, B is a Formal Concept iff A contains only objects sharing all attributes from B and B contains just the attributes shared by all members of A. Tables 2.2 and 2.3 show two formal concepts derivable from the formal context of Table 2.1.

Formal Concepts are ordered using a *subconcept-superconcept* relation. In our example, superconcepts of *edible liquids* are the *edibles* $\langle \{Lollypop, Whiskey, Ice -$

Table 2.3: Example of *formal concept*. The concept *intension* is the set $\{Liquid, Edible\}$, while the *extension* is $\{Whiskey, Ice - Cream, Coffee\}$.

-	Liquid	Edible	Solid
Drone			x
Lollypop		x	x
Watch			x
Ring			x
Shoes			x
Detergent	x		
Whiskey	x	x	
Ice-Cream	x	x	x
Coffee	x	x	

$Cream, Coffee\}, \{Edible\}\rangle$ and the *liquids* $\langle\{Detergent, Whiskey, Ice - Cream, Coffee\}, \{Liquid\}\rangle$. Similarly, a subconcept of *wearable solids* is *pocketable, wearable solids* $\langle\{Hat, Ring, Watch\}, \{pocketable, Wearable, Solid\}\rangle$.

This relation is defined as follows:

Definition 2.3 (Subconcept-superconcept ordering: $\leq$) *For two concepts $\langle A_1, B_1\rangle$ and $\langle A_2, B_2\rangle$, both member of $\langle X, Y, I\rangle$, $\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle$ iff $A_1 \subseteq A_2$ (iff $B_2 \subseteq B_1$)*

The symbol $\leq$ represents a partial order, meaning that the concept $\langle A_1, B_1\rangle$ is more specific then $\langle A_2, B_2\rangle$ (and counter-wise $\langle A_2, B_2\rangle$ is more general).

The collection of all Formal Concepts of a given Formal Context is called *Concept Lattice*, and it is defined as the association between a Formal Context and the previously defined ordering operator.

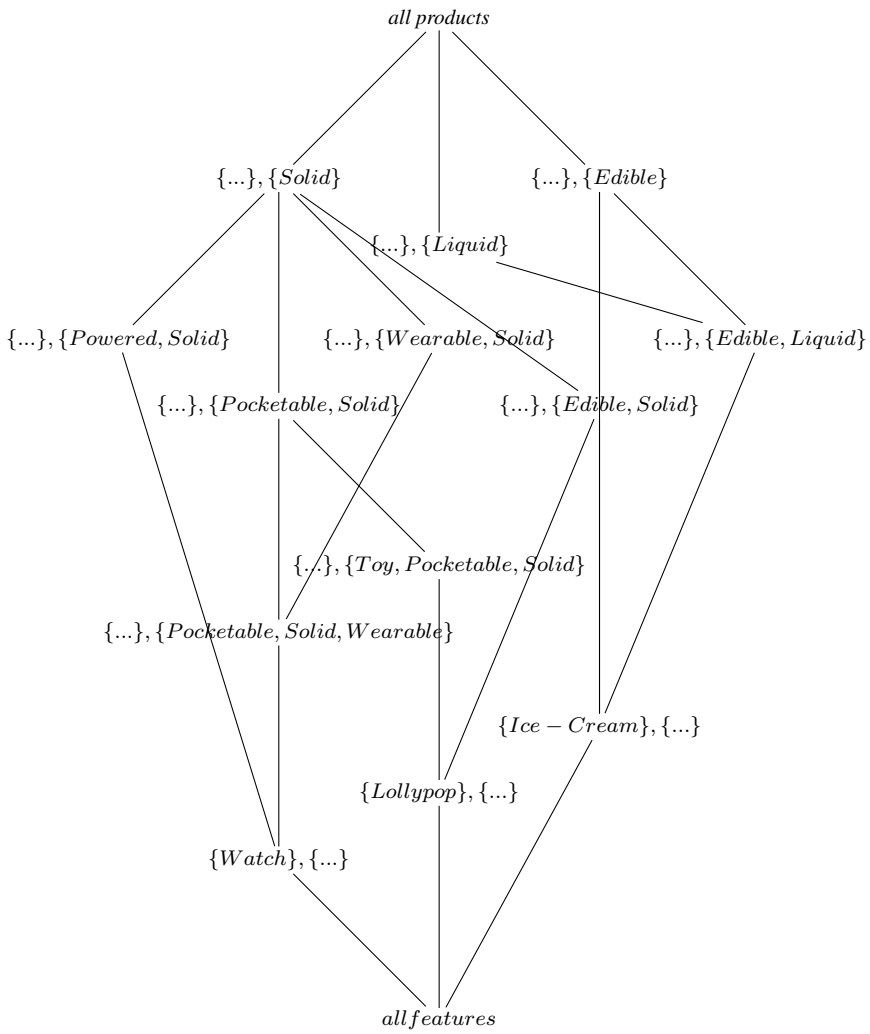
Definition 2.4 (Concept Lattice) *$C(X, Y, I)$ is the collection of all formal concepts of the context $\langle X, Y, I\rangle$, therefore:*

$$C(X, Y, I) = \{\langle A, B\rangle \ni 2^X, 2^Y | A' = B, B' = A\}$$

i.e. A is a member of the powerset of objects and B is a member of the powerset of attributes, and for each object set A in 2^X there exists an attribute set B in 2^Y such that $A' = B$ and $B' = A$.

This definition reflects a fundamental property of *Galois connections*, where our *closure* operator $'$ is represented as a pair of monotonic functions, which establish a biunivocal relation between the two sets. Figure 2.2 shows the lattice resulting from the formal context of Table 2.1. At the top of the lattice is the concept of all products, with an empty intent. Conversely, at the bottom is the concept of all features, with an empty extent. These two origins of the lattice are called the *Supremum* and the *Infimum* concepts, and are defined as follows:

Figure 2.2: Concept Lattice



Definition 2.5 (Infimum) *The bottom node of the Concept Lattice, being the Concept with the smallest extent (and largest intent).*

Definition 2.6 (Supremum) *The top node of the Concept Lattice, being the Concept with the smallest intent (and largest extent).*

Between the two extremes of the lattice, all other nodes (concepts) are connected through the ordering relation $\leq$ (see Figure 2.2).

Research on FCA and its applications include several aspects. The generation of the concept set and the ordering operation are problematic in terms of complexity and many algorithms have been designed to compute lattices, already prior to the development of FCA as a discipline. The

solutions vary in many details, but they usually rely on two general approaches. A *waterfall* approach generates all concept sets and then performs the lattice construction (ordering) (for example Chein [Chein (1969)]), while an *incremental* approach creates and modifies a lattice on the go (for example Godin [Godin et al. (1995)]). A comparative description of the properties of classic FCA algorithms is in [Kuznetsov and Obiedkov (2002)]. An interesting aspect of FCA applications is related to user interaction, referring to the scalability and usability of interfaces, and to *sense-making*. This last aspect becomes problematic when the size of the concept set increases, particularly in the presence of sparse data, when very few attributes are shared among a small number of objects. FCA is also a *clustering technique*, and shares with other Data Mining approaches the problem of *explanation*, also known as concept *labelling*. Indeed, assigning a meaningful, descriptive and informative name to each concept in the lattice can be a tedious and difficult task. Therefore, approaches to assist the user have been developed in advanced interfaces. Other approaches modify the lattice by removing concepts considered less informative, depending on the size of the extent or intent. These approaches rarely generalize outside specific domains or use cases. Another common area of application of FCA is for association rule mining, for example to generate recommendations. FCA has been also studied in the context of Information Retrieval (IR) [Cimiano et al. (2005)] and Knowledge Discovery [Codocedo and Napoli (2015); Poelmans et al. (2010)]. Two recent surveys illustrate advances in techniques and applications of FCA [Poelmans et al. (2013a,b)].

Formal Concept Analysis as a research area is not directly related to Knowledge Engineering. However FCA received interest as a classification method in particular in approaches for bottom-up ontology construction [Cimiano et al. (2004); Obitko et al. (2004); Van Der Vet and Mars (1998)], and more recently in Semantic Web and Linked Data research [Alam and Napoli (2015); Ferré (2015)].

2.4 Semantic Web Technologies

The Semantic Web is an extension of the WWW through a set of standards developed by the W3C targeted at allowing machines to autonomously interact by exchanging data. The standards include common protocols and data formats that allow information to be shared across systems, enterprise and domain boundaries. Semantic Web technologies (RDF, OWL, SKOS, SPARQL, etc.) constitute a framework where Web applications can query the data and draw inferences based on shared vocabularies. The objective of this section is to introduce the fundamental concepts of Semantic Web technologies, from which we inherit the vision, but also as a set of technologies and formalisms that are extensively used in our work.

As already mentioned in Section 2.2, researchers in Knowledge Engineering made the idea of the Semantic Web emerge, therefore a variety of solutions produced by the latter can be considered as being strongly influenced by the former. It is the case of the fundamental distinction between syntax and semantics, which goes beyond the requirement of providing a machine-readable version of the content of Web pages, and resembles Newell's distinction between the symbol level and the knowledge level. This notion is at the basis of the Resource Description Framework (RDF), conceived as an abstract, generic and domain independent *meta-model*. The intersection between the field of Ontology Engineering and Semantic Web is also significant at least in the sub-area related to Web Ontologies, as testified by the extensive research on Web Reasoning, Networked Ontologies, and Ontology Design Patterns. The Semantic Web framework is structured as a layered architecture

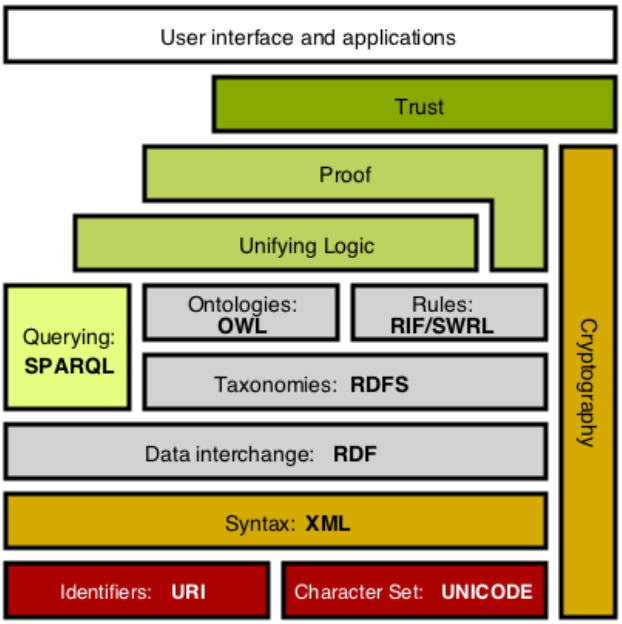


Figure 2.3: The Semantic Web Stack[1].

(see Figure 2.3). At the bottom of the stack, there are technologies that form also the basis of the hypertext Web, these are Uniform Resource Identifiers (URI), Unicode and the family of XML related technologies. (However, syntax formats other then XML are also used, like Turtle, or more recently the popular JavaScript Object Notation (JSON) and its extension JSON-LD.) Following the principles of Knowledge Engineering, we can call this the *symbol* level, where what is important is how symbols are represented and shared in a common serialisable model. The foundation layer of the Semantic Web is the Resource Description Framework (RDF), a generic meta-model that allows the exchange of data by the means of a minimal graph-based structure. To allow a richer

[1]Figure taken from `https://en.wikipedia.org/wiki/Semantic_Web_Stack`, accessed 26/01/2017.

representation, RDF Schema (RDFS) defines a set of language primitives for the formalisation of basic schemas in RDF. On top of that, specifications like Simple Knowledge Object System (SKOS) and the Web Ontology Language (OWL) permit on the one hand to describe taxonomies of concepts and their relations (SKOS), and on the other hand to express logical constraints on the content of RDF graphs (OWL). With these set of technologies, systems are capable of publishing information on the WWW as Linked Data (LD), consuming this information with the SPARQL query language, and derive inferences by exploiting implicit knowledge derived by the ontologies or vocabularies used.

We provide now a background view on the Semantic Web, as we will adopt a combination of these technologies in our contribution.

2.4.1 Resource Description Framework (RDF)

RDF is a generic meta-model based on a graph like structure made of labelled nodes and arcs. The basic structure is the *triple*, which includes a subject node, a labelled directed arc called predicate, and a target object node (see Figure2.4). Each one of the components of a triple is an element

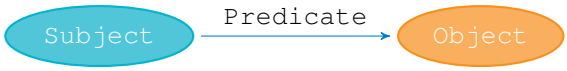


Figure 2.4: RDF triple.

of an RDF *statement*. For instance, Figure 2.5 illustrates the triple representing the statement `studies-at(enridaga,OU)`. However, using such short strings as names is not convenient,

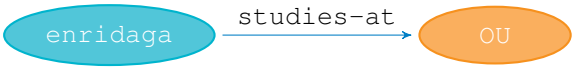


Figure 2.5: Example of RDF triple.

particularly if we want to identify the elements of RDF statements on the Web. Precisely, an RDF *term* can be of one of three types:

– Internationalized Resource Identifiers (IRI): IRI is a Unicode string [Unicode Consortium and others (2004)] that conforms to the syntax defined in RFC 3987 [Duerst and Suignard (2005)]. It is essentially a URI [Berners-Lee et al. (2005)] allowing a larger set of characters, therefore including symbols outside the English alphabet. The role of IRIs is the one of being

portable between different RDF datasets. IRIs can be used to name subjects and objects in RDF statements, and they *must* be used to name predicates.

– Literal: a literal is a value associated with an IRI representing its datatype, or a language annotation (also called *tag*, it is a two digit string following the ISO 3166-2 specification [ISO (1998)]). Some examples include:
  - `"5"^^<http://www.w3.org/2001/XMLSchema#int>`
  - `"Rome is located in Italy"@en.`
  - `"0.5"^^<http://www.w3.org/2001/XMLSchema#float>`
  - `"OU"^^<http://www.w3.org/2001/XMLSchema#string>`
  - `"RGF2dmVybyB0aSBtZXR0aSBhIGZhcmUgdW5hIGNvc2EgZG VsIGdlb-mVyZT8="`
    `^^<http://www.w3.org/2001/XMLSchema#base64>`
  - `"Rome"`, assuming the default datatype `http://www.w3.org/2001/XMLSchema#string`.

  Literals can only be used as objects in RDF statements.
– Blank node. A blank node is an anonymous element, without a portable identifier. In RDF statements, it can be used as subject or object, but never as a predicate.

Here is a list of RDF statements, also known as RDF *graph*, in N-Triples[2] syntax:

```
<http://www.example.org/person/enridaga> <http://www.example.org/property/
   studies-at> <http://www.example.org/organization/OU> .
<http://www.example.org/person/enridaga> <http://www.example.org/property/
   name> "Enrico Daga"^^<http://www.w3.org/2001/XMLSchema#string> .
<http://www.example.org/person/enridaga> <http://www.example.org/property/
   has-child> _:x000 .
_:x000 <http://www.example.org/property/name> "Filippo"^^<http://www.w3.
   org/2001/XMLSchema#string> .
<http://www.example.org/person/enridaga> <http://www.example.org/property/
   has-child> _:x001 .
_:x001 <http://www.example.org/property/name> "Pietro"^^<http://www.w3.org
   /2001/XMLSchema#string> .
```

The symbols `_:x001` and `_:x002` are blank nodes, which are local identifiers, prepended in N-Triples by the prefix `_:`. This verbose representation can be reduced by using some forms of abbreviation, supported by other types of syntax, for example, the Turtle syntax. The notion of *namespace* is already present in XML and related languages and derives from the application of URIs/IRIs as a naming system. In computer science, a namespace is a collection of symbols

[2]RDF 1.1 N-Triples, `http://www.w3.org/TR/2014/REC-n-triples-20140225/`.

meant to organise items of different kinds so that they can be identified by a name. Usually, and it is the case of URIs/IRIs, they are constituted by an *authority* part, and by a *naming* part. XML Namespaces are URIs ending with "/" or "#" and their purpose is to avoid naming clashes between XML documents produced by different authorities. In document serialisation, namespaces are associated with so-called *prefixes*, which will be used as readable placeholders instead of the fully qualified name. The triple set listed above can be serialised in Turtle as follows:

```
prefix per: <http://www.example.org/person/> .
prefix pro: <http://www.example.org/property/> .
prefix org: <http://www.example.org/organization/> .
prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

per:enridaga pro:studies-at org:OU ;
             pro:name "Enrico Daga"^^xsd:string ;
             pro:has-child
               [ pro:name "Filippo"^^xsd:string ],
               [ pro:name "Pietro"^^xsd:string ] .
```

However, in real-world scenarios namespaces reflect the authorities establishing the actual name, identified by the domain part of the URI. A real-world example is the following RDF document, extracted from `http://data.open.ac.uk` [Daga et al. (2016b)], the Linked Open Data endpoint of the Open University:

Listing 2.1: An RDF graph serialized in Turtle.

```
prefix per: <http://data.open.ac.uk/person/> .
prefix foaf: <http://xmlns.com/foaf/0.1/> .
prefix dcterm: <http://purl.org/dc/terms/> .
prefix tag: <http://ns.inria.fr/nicetag/2010/09/09/voc#> .
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
prefix void: <<http://rdfs.org/ns/void#> .
prefix schema: <http://schema.org/> .
prefix vivo: <http://vivoweb.org/ontology/core#> .
prefix part: <http://purl.org/vocab/participation/schema#> .
prefix role: <http://data.open.ac.uk/role/> .
prefix org: <http://www.w3.org/ns/org#> .


<http://data.open.ac.uk/resource/person/0fe416343d163a372e32910118bdbe76>
        a foaf:Document ;
        foaf:primaryTopic per:0fe416343d163a372e32910118bdbe76 .

per:0fe416343d163a372e32910118bdbe76
        a foaf:Person ;
        rdfs:label "Enrico Daga"@en ;
```

```
        dcterms: description
            "<p>I am Project Officer - Linked Data  at the Knowledge
               Media Institute of The Open University. My main role is the
                maintenance and evolution of<a href=\"http://data.open.ac.
               uk\">http://data.open.ac.uk</a>.</p><p>I was previously
               employed at the Italian National Research Council as&quot;
               Technology Expert&quot;. In the past years, I participated
               to the following research projects: European NeOn Project (
                www.neon-project.org ), and Interactive Knowledge
               Stack (IKS) project (<a href=\"http://www.iks-project.org\"
                rel=\"nofollow\">www.iks-project.org</a>).</p><p>I work on
                the MK:Smart project (<a href=\"http://www.mksmart.org\">
               www.mksmart.org/</a>) on data cataloguing and data
               governance.</p><p> </p>"^^rdf:HTML ;
        part:holder_of
           role:AcademicRelatedStaff ;
        schema:jobTitle  "Project Officer: Linked Data"@en ;
        org:memberOf
           <http://data.open.ac.uk/organization/kmi> ;
        foaf:topic_interest
           <http://data.open.ac.uk/topic/web_science> ,
           <http://data.open.ac.uk/theme/software_engineering_and_design>
              ,
           <http://data.open.ac.uk/topic/semantics_and_ontologies> ;
        foaf:weblog <http://www.enridaga.net> ;
        foaf:workInfoHomepage
            <http://www.open.ac.uk/people/ed4565> ,
            <http://kmi.open.ac.uk/people/member/enrico-daga> .
```

The above listing is the content of an RDF document, representing the graph depicted in Figure 2.6. The keyword `a` is equivalent to `rdf:type` in Turtle (and also SPARQL).

The main purpose of RDF and related technologies is to represent information on the WWW. RDF graphs are meant to be portable artefacts. Therefore it is useful to have a method to identify an RDF graph univocally. RDF 1.1 [Lanthaler et al. (2014)] introduces Named Graphs, recommending to add a fourth dimension to the RDF statement, declaring the *context* of it with a proper IRI. For example, one might assign to the triples of the RDF graph of Figure 2.6 the Web address of the document containing it as IRI (Uniform Resource Locator (URL) is a URI, URIs are IRIs ...). RDF databases (like Apache Jena TDB[3], OpenLink Virtuoso[4] or AllegroGraph[5], just to mention a few examples) allow to manage and query RDF statements following this quad-based model. Specific formats can be used to exchange quad data: N-Quads[6], TriX (Named Graphs in XML), and TriG (Named Graphs in Turtle). Named Graphs have an important role when querying RDF databases

[3]Apache Jena TDB: `https://jena.apache.org/documentation/tdb/`

[4]OpenLink Virtuoso: `https://virtuoso.openlinksw.com/`

[5]AllegroGraph: `https://allegrograph.com/`

[6]N-Quads. `https://www.w3.org/TR/n-quads/`

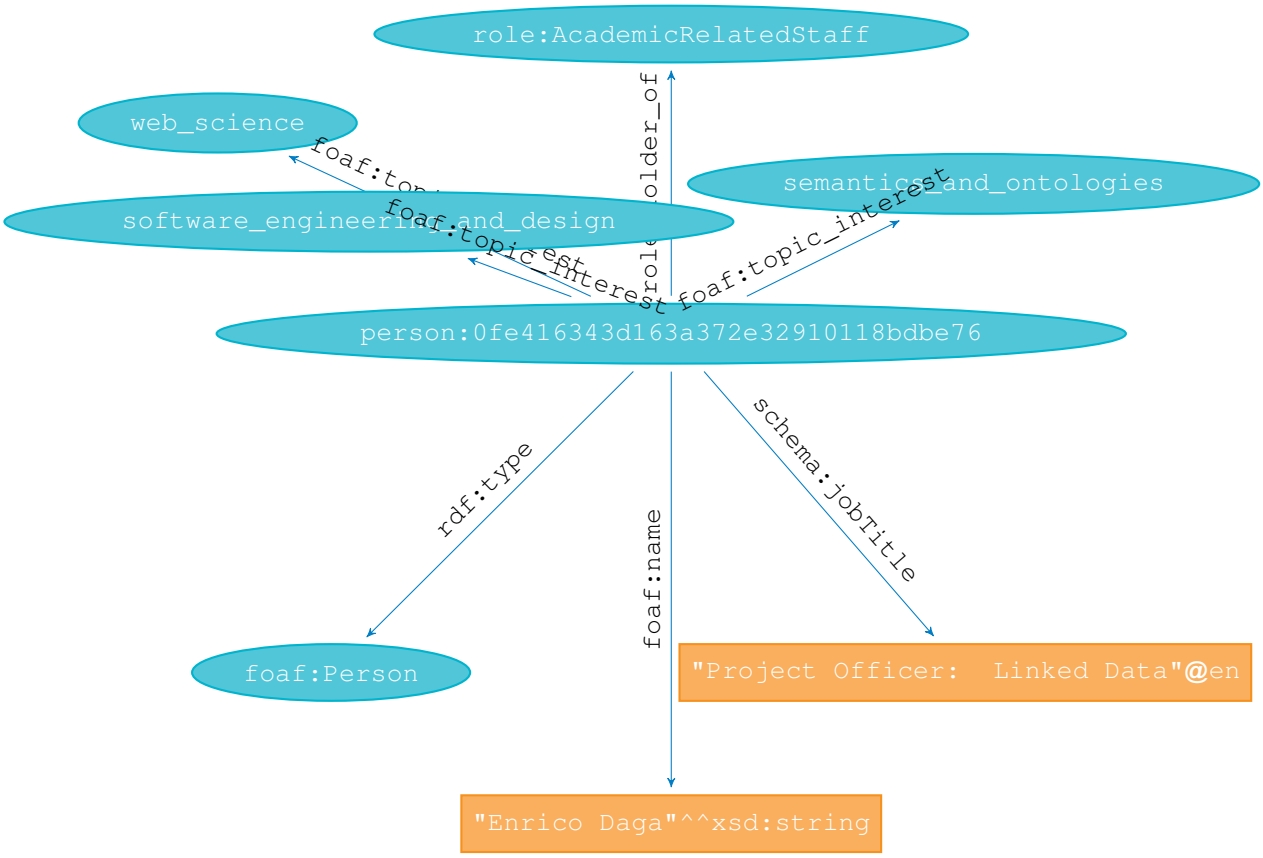


Figure 2.6: RDF graph, graphical visualization.

with SPARQL.

2.4.2 Publishing and consuming Semantic Web data

RDF has been designed with the purpose to provide a general model for data representation. As a formalism, it does specify very little about the nature of the content to be represented, limiting itself to the structure for representing it.

Publishing data on the Semantic Web requires the use of shared models, or vocabularies, which allows us to define univocally the meaning of the information contained in the datasets. Vocabularies can be described using a set of primitives, declared in the RDF Schema (RDFS) specification [Guha and Brickley (2014)]. These primitives are:

- `rdfs:Resource` - the class of anything being referred by a IRI, a blank node, or being a data value.
- `rdfs:Class` - the class of all classes. For example `foaf:Person` is an `rdfs:Class`.
- `rdfs:Literal` - the class of data values.

- `rdfs:Datatype` - the class of literal datatypes. For example `xsd:int` is a `rdfs:Datatype`.
- `rdf:Property` - the class of properties. For example `foaf:name` is a `rdf:Property`.
- `rdf:XMLLiteral` - the class of literals of type xml. In other words, the datatype of XML literals:

```
rdf:XMLLiteral
   rdf:type rdfs:Datatype ;
   rdfs:subClassOf rdfs:Literal .
```

- `rdf:type` - the property that assigns an rdfs:Class to a resource.
- `rdfs:label` - assigns a literal as human readable placeholder for the resource.
- `rdfs:comment` - annotate the resource with a comment.
- `rdfs:subClassOf` - links a subclass to a superclass. For example `foaf:Person rdfs:subClassOf foaf:Agent`.
- `rdfs:subPropertyOf` - links a property to a super property. For example `skos:prefLabel rdfs:subPropertyOf rdfs:label`.
- `rdfs:domain` - specifies the type of the source of a property. For example:

```
dcat:downloadURL
    rdf:type rdf:Property ;
    rdfs:range rdfs:Resource .
```

- `rdfs:range` - similarly, specifies the type of the target of a property. For example:

```
dcat:downloadURL
    rdf:type rdf:Property ;
    rdfs:domain dcat:Distribution .
```

- `rdfs:seeAlso` - annotates the resource linking it to another resource of interest.
- `rdfs:isDefinedBy` - connects a resource to the resource defining it. It can be used to link a class to the document where it is described.

(We leave out other primitive vocabulary elements, like containers, collections or reification, as we will not make use of them in this work.)

Vocabularies are descriptions of classes and properties to be used when developing RDF data on the Web. The value of these descriptions is in the fact that they are (a) shared conceptualisations that can be used by different systems independently and in communication, (b) openly available on the Web, therefore the meaning of each property and class is explicit, and can be acquired from the data by following its IRI, (c) usable in combination, so that data publishers can select and pick the terms from several vocabularies to define new data models to fit their needs. Vocabularies

are generally developed to fit specific use cases. The Friend of a Friend (`foaf`) vocabulary was developed to describe persons and their social networks on the Web, and the Simple Knowledge Object System (`skos`) to support the development of conceptual taxonomies like the ones used to organize media objects on content management systems or to publish scientific terminology in RDF (like thesauri in the library domain). Within the Dublic Core Metadata Initiative (DCMI) is maintained the definition of a set of terms widely used across many domains such as `dc:creator` or `dc:publisher`, just to mention two prominent examples. The Linked Data (LD) promise of developing an interlinked Web of Data to be query-able as a giant distributed database is based on shared vocabularies to represent any type of knowledge. Therefore vocabularies have different levels of sophistication. The Creative Commons Rights Expression language (`cc`) includes terms necessary to link resources to Creative Commons licenses on the Web. With the Time Ontology (`time`) [Cox and Little (2016)] it is possible to express temporal entities like dates, time points and time sequences and intervals. The RDF Data Cube Vocabulary [Reynolds and Cyganiak (2014a)] has been developed to support the publishing of multidimensional data, a common information metamodel particularly in the domain of statistic.

However, shared vocabularies are only a part of data understanding and reuse. An equally important one is the capability of linking resources between datasets. In the Linked Data, resources are *linked* from a dataset to another one by the means of the `owl:sameAs` property. This technique is at the core of the Linked Data infrastructure. As a result, an important role is taken by some public databases as naming entity systems. It is the case of DBPedia, a large database generated from Wikipedia that is one of the *hubs* of Linked Data. Geonames, similarly, publishes a list of geographical toponyms that can be directly used or linked. Shared identifiers are crucial as much as shared schemas for content reuse. For example, Named Entity Recognition systems can be deployed to link textual content (like web pages) and data.

The W3C developed a set of guidelines for Linked Data publishers [Atemezing et al. (2014)]:

> STEP #1 PREPARE STAKEHOLDERS: Prepare stakeholders by explaining the process of creating and maintaining Linked Open Data.
>
> STEP #2 SELECT A DATASET: Select a dataset that provides benefit to others for reuse.
>
> STEP #3 MODEL THE DATA: modelling Linked Data involves representing data objects and how they are related in an application-independent way.
>
> STEP #4 SPECIFY AN APPROPRIATE LICENSE: Specify an appropriate open data license. Data reuse is more likely to occur when there is a clear statement about the origin, ownership, and terms related to the use of the published data.

STEP #5 GOOD URIs FOR LINKED DATA: The core of Linked Data is a well-considered URI naming strategy and implementation plan, based on HTTP URIs. Consideration for naming objects, multilingual support, data change over time and persistence strategy are the building blocks for useful Linked Data.

STEP #6 USE STANDARD VOCABULARIES: Describe objects with previously defined vocabularies whenever possible. Extend standard vocabularies where necessary, and create vocabularies (only when required) that follow best practices whenever possible.

STEP #7 CONVERT DATA: Convert data to a Linked Data representation. This is typically done by a script or other automated processes.

STEP #8 PROVIDE MACHINE ACCESS TO DATA: Provide various ways for search engines and other automated processes to access data using standard Web mechanisms.

STEP #9 ANNOUNCE NEW DATA SETS: Remember to announce new data sets on an authoritative domain. Importantly, remember that as a Linked Open Data publisher, an implicit social contract is in effect.

STEP #10 RECOGNIZE THE SOCIAL CONTRACT: Recognize your responsibility in maintaining data once it is published. Ensure that the dataset(s) remain available where your organization says it will be and is maintained over time. [Atemezing et al. (2014)]

The Linked Open Data (LOD) is not only made of data and machines exchanging them but also on the practices and developers involved in maintaining it. These include services in support of whoever wants to interact with Linked Data as publisher or consumer. Some of them are:

1. http://datahub.io - the Open Knowledge Foundation catalogue of open datasets
2. http://lov.okfn.org/ - the Linked Open Vocabularies project, to find and choose vocabulary terms
3. http://yasgui.org/ - A user interface to query SPARQL endpoints
4. http://sameas.org/ - To find co-references between different datasets
5. http://prefix.cc - A service to obtain popular namespaces and their prefixes
6. http://lodlaundromat.org/ - A database made of the crawling and cleaning of the LOD

Examples of Linked Open Datasets that have a central position in the LOD cloud because of their role of named entity systems are:

1. http://dbpedia.org - Wikipedia entities as Linked Data

2. http://www.geonames.org/ - a geographical database with millions of place names
3. http://sparql.europeana.eu/ - European Cultural Heritage data as LD

Semantic Web Data can be published in various ways, from downloadable files to embedded annotations in HTML (using techniques such as RDFa[7] and vocabularies such as `http://schema.org/`). The *follow-your-nose* approach is based on dereferencing IRIs mentioned in the RDF datasets in order to find related information. This method, also called graph traversal, enables the discovering of new data in a similar way a user navigates the web following links in HTML pages. HTTP content negotiation allows programs to request the data in a specific seralisation format, choosing from the many available for RDF: RDF/XML, Turtle, JSON-LD, N-Triples, Trix, and so on[8]. Probably the primary method to consume RDF datasets is the SPARQL Protocol and Query language.

2.4.3 SPARQL Protocol and Query Language

The "SPARQL Protocol and Query language" (SPARQL) is a family of W3C specifications for enabling Web systems to access, query and modify RDF datasets. SPARQL includes:

- A query language, based on a graph matching approach.
- An update language, to manipulate RDF Datasets.
- A service description vocabulary, in order to describe SPARQL services in RDF.
- A federated querying system.
- Query result syntax for tabular views, in XML and JSON formats.
- Entailment regimes, defining how SPARQL query execution engines can produce inferred knowledge by exploiting other specifications such as RDFS, OWL or RIF (in what follows we assume queries to be interpreted within a Simple Entailment regime).
- A query protocol that describes the behaviour of a so-called SPARQL *endpoint*. A SPARQL endpoint is a Web API meant to interact with an RDF Dataset.
- A graph store protocol, which is a CRUD API to enable direct access to RDF graphs following the REST principles.

SPARQL is used several times in our work. In what follows we focus on the basic characteristics of the query language, therefore only covering the way RDF data can be read with SPARQL. The reader can consult the W3C documents for the other parts of the specifications and for details of the language we are leaving out [The W3C SPARQL Working Group (2013)].

[7] RDFa: `https://www.w3.org/TR/2015/NOTE-rdfa-primer-20150317/`
[8] W3C RDF home page: `https://www.w3.org/RDF/`

The SPARQL language is based on a graph matching approach. At the core is the notion of triple pattern. A triple pattern is like an RDF triple where the subject, predicate or object can be variable terms. A triple pattern matches a portion of an RDF graph where there are RDF terms from that graph that can substitute the variable terms in the given position. The resulting RDF graph will, therefore, be equivalent to the matched subgraph. Taking as example the RDF graph in Listing 2.1, the triple pattern

```
?x rdf:type foaf:Person
```

would actually match the triple

```
per:0fe416343d163a372e32910118bdbe76 rdf:type foaf:Person
```

while the triple pattern

```
<http://data.open.ac.uk/organization/kmi> ?y ?z
```

would not have matches, as there is no triple with the RDF term `<http://data.open.ac.uk/organization/kmi>` in the subject position.

A set of triple patterns is called a Basic Graph Pattern (BGP). A BGP will have a *solution sequence* according to the way it matches the RDF graph, therefore there can be zero, one, or multiple solutions to a given query. For example, the pattern

```
?x rdf:type foaf:Person .
?x rdfs:label ?y
```

will have a single solution:

```
?x=per:0fe416343d163a372e32910118bdbe76
?y="Enrico Daga"@en
```

while the pattern

```
?x rdf:type foaf:Person .
?x foaf:workInfoHomepage ?y
```

will have actually two solutions:

```
?x=per:0fe416343d163a372e32910118bdbe76
?y=<http://www.open.ac.uk/people/ed4565>

?x=per:0fe416343d163a372e32910118bdbe76
?y=<http://kmi.open.ac.uk/people/member/enrico-daga>
```

The last two BGPs are examples of a *conjunction* of triple patterns. It is worth noting that all the variables used in graph patterns must be *bound* in every solution. Similarly, it is possible to design queries on RDF datasets by extending the BGP notion to also include named graphs. Multigraph datasets can be represented in N-Quads syntax [Carothers (2014)]. N-Quads follows the same principle of N-Triples, but includes an additional IRI at the beginning of the tuple, identifying the dataset containing the triple. Considering the RDF Dataset in Listing 2.2:

Listing 2.2: An example RDF Dataset in N-Quads syntax.

```
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/0fe416343d163a372e32910118bdbe76> <http://www.w3.org/1999/02/22-
   rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/0fe416343d163a372e32910118bdbe76> org:memberOf <http://data.open
   .ac.uk/organization/kmi> .
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/d9734c68df46924452ff25a4174ab758> <http://www.w3.org/1999/02/22-
   rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/d9734c68df46924452ff25a4174ab758> <http://xmlns.com/foaf/0.1/
   topic_interest> <http://data.open.ac.uk/topic/web_science> .
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/d9734c68df46924452ff25a4174ab758> <http://xmlns.com/foaf/0.1/
   topic_interest> <http://data.open.ac.uk/theme/
   software_engineering_and_design> .
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/d9734c68df46924452ff25a4174ab758> <http://xmlns.com/foaf/0.1/
   topic_interest> <http://data.open.ac.uk/topic/semantics_and_ontologies>
    .
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/d9734c68df46924452ff25a4174ab758> <http://xmlns.com/foaf/0.1/
   weblog> <http://www.enridaga.net> .
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/d9734c68df46924452ff25a4174ab758> <http://xmlns.com/foaf/0.1/
   workInfoHomepage> <http://www.open.ac.uk/people/ed4565> .
<http://data.open.ac.uk/context/people/profiles> <http://data.open.ac.uk/
   person/d9734c68df46924452ff25a4174ab758> <http://xmlns.com/foaf/0.1/
   workInfoHomepage> <http://kmi.open.ac.uk/people/member/enrico-daga> .
<http://data.open.ac.uk/context/oro> <http://data.open.ac.uk/oro/42047> <
   http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/
   ontology/bibo/AcademicArticle> .
<http://data.open.ac.uk/context/oro> <http://data.open.ac.uk/oro/42047> <
   http://www.w3.org/2000/01/rdf-schema#label> "Dealing with diversity in
   a smart-city datahub"
<http://data.open.ac.uk/context/oro> <http://data.open.ac.uk/oro/42047> <
   http://purl.org/dc/elements/1.1/creator> <http://data.open.ac.uk/person
   /0fe416343d163a372e32910118bdbe76> .
<http://data.open.ac.uk/context/oro> <http://data.open.ac.uk/oro/42047> <
   http://purl.org/dc/elements/1.1/creator> <http://data.open.ac.uk/person
```

```
   /0e5d4257051894026ea74b7ed55557e7> .
<http://data.open.ac.uk/context/oro> <http://data.open.ac.uk/oro/41070> <
   http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/
   ontology/bibo/Book> .
<http://data.open.ac.uk/context/oro> <http://data.open.ac.uk/oro/41070> <
   http://www.w3.org/2000/01/rdf-schema#label> "Educational Technology
   Topic Guide" .
<http://data.open.ac.uk/context/oro> <http://data.open.ac.uk/oro/41070> <
   http://purl.org/dc/elements/1.1/creator> <http://data.open.ac.uk/person
   /d9734c68df46924452ff25a4174ab758> .
```

and the graph pattern

```
prefix [...]

graph <http://data.open.ac.uk/context/people/profiles> {
    ?x org:memberOf <http://data.open.ac.uk/organization/kmi>
} .
graph <http://data.open.ac.uk/context/oro> {
        ?y dc:creator ?x
}
```

will have the following solution

```
?x=per:0fe416343d163a372e32910118bdbe76
?y=<http://data.open.ac.uk/oro/42047>
```

The SPARQL query language provides four methods for handling query solutions or query *types*: `ASK`, `SELECT`, `CONSTRUCT`, and `DESCRIBE`. A `ASK` type of query can return a boolean value, being `true` when the graph pattern has at least one solution, `false` otherwise. For example, the following query over the RDF Dataset in Listing 2.2 will return `true`:

```
prefix [...]

ASK {
 ?x rdf:type foaf:Person .
 ?x foaf:workInfoHomepage []
}
```

In the previous example, the symbol `[]` represents a blank node. (In graph patterns, blank nodes act as existential identifiers, therefore any RDF term would match it.)

`SELECT` queries return the query solutions projected as tabular data. The following query:

```
prefix [...]

SELECT ?y WHERE {
 ?x rdf:type foaf:Person .
```

```
 ?x foaf:topic_interest ?y
}
```

will have the following matching solutions:

```
?x=per:0fe416343d163a372e32910118bdbe76
?y=<http://data.open.ac.uk/topic/web_science>

?x=per:0fe416343d163a372e32910118bdbe76
?y=<http://data.open.ac.uk/theme/software_engineering_and_design>

?x=per:0fe416343d163a372e32910118bdbe76
?y=<http://data.open.ac.uk/topic/semantics_and_ontologies>
```

but only one variable will be *projected* to the result set:

```
-------------------------------------------------------------------
| y                                                               |
-------------------------------------------------------------------
|<http://data.open.ac.uk/topic/web_science>                       |
|<http://data.open.ac.uk/theme/software_engineering_and_design>|
|<http://data.open.ac.uk/topic/semantics_and_ontologies>          |
-------------------------------------------------------------------
```

The `SELECT` query supports also *aggregate* functions (COUNT, SUM, AVG, MIN, MAX, ...), for example:

```
prefix ...

SELECT ?x (COUNT(?y) AS ?topics) WHERE {
 ?x rdf:type foaf:Person .
 ?x foaf:topic_interest ?y
} GROUP BY ?x
```

returning:

```
-----------------------------------------------
| x                                   | topics |
-----------------------------------------------
|per:0fe416343d163a372e32910118bdbe76 | 3      |
-----------------------------------------------
```

SPARQL can also be used to generate a new graph out of the variable projections of a graph pattern towards an RDF dataset. This is done using the `CONSTRUCT` query type:

```
prefix ...
prefix dbowl: <http://dbpedia.org/ontology/>
```

```
CONSTRUCT {
 ?y rdf:type dbowl:topic .
 ?org rdf:type org:Organization .
 ?org foaf:topic_interest ?y
} WHERE {
 ?x foaf:topic_interest ?y .
 ?x org:memberOf ?org
}
```

that, executed over the RDF dataset in Listing 2.2 will produce the following RDF graph:

```
<http://data.open.ac.uk/topic/web_science> rdf:type dbowl:topic .
<http://data.open.ac.uk/topic/semantics_and_ontologies> rdf:type dbowl:
   topic .
<http://data.open.ac.uk/theme/software_engineering_and_design> rdf:type
   dbowl:topic .
<http://data.open.ac.uk/organization/kmi> rdf:type org:Organization .
<http://data.open.ac.uk/organization/kmi> foaf:topic_interest <http://data
   .open.ac.uk/topic/web_science> .
<http://data.open.ac.uk/organization/kmi> foaf:topic_interest <http://data
   .open.ac.uk/topic/semantics_and_ontologies> .
<http://data.open.ac.uk/organization/kmi> foaf:topic_interest <http://data
   .open.ac.uk/theme/software_engineering_and_design> .
```

Finally, the `DESCRIBE` query type requests to return the RDF subgraph *about* the given terms or projected variables. Examples of DESCRIBE queries are the following:

```
DESCRIBE <http://data.open.ac.uk/topic/web_science>
```

```
DESCRIBE ?org WHERE {
 ?x foaf:topic_interest <http://data.open.ac.uk/topic/web_science> .
 ?x org:memberOf ?org
}
```

The returning graphs would include all the triples having the declared term or projected values as subject (however, the semantics of DESRIBE may vary between implementations, some of them returning also triples having the term in the object position).

The SPARQL query language includes many features not mentioned here. These include RDF term manipulation functions, string manipulation functions, basic mathematical functions, boolean and conditional operators. Moreover, Basic Graph Patterns can be extended with sophisticated constructs allowing fine-grained matching criteria using operators such as `FILTER( ... )` or `OPTIONAL {...}`, including existential qualifiers - `FILTER EXISTS { ... }`, union of query solutions - `{ ... } UNION { ... }`, negative matching - `FILTER NOT EXISTS { ... }`. SPARQL also supports graph traversal constructs (so-called property paths). Query

nesting is another useful feature, and it is the basis for supporting query federation through the `SERVICE` clause. Some implementations also extend SPARQL with additional features, for example adding operators and datatypes for handling *geospatial* data [Battle and Kolas (2012)]. For more details on the language, we refer to the W3C specification [The W3C SPARQL Working Group (2013)].

2.4.4 Reasoning with RDF data

RDF models make use of classes and properties which meaning is specified in shared vocabularies. Fixing the semantics of RDF terms is important to support a consistent usage between datasets. An equally important aspect is related to the support for automated reasoning. By the means of RDFS schema - and other languages built on top of it - computer programs are capable of producing *inferred* knowledge from the triples *asserted* in consumed RDF datasets.

The basic inferences can be derived from RDFS entailments. RDFS entailment can be expressed as Horn clauses:

```
subClassOf(x,z) ← subClassOf(x,y)  ∧  subClassOf(y,z)
subPropertyOf(a,c) ← subPropertyOf(a,b) ∧ subPropertyOf(b,c) .
type(z,y) ← domain(x,y), triple(z,x,q) .
type(q,y) ← range(x,y), triple(z,x,q) .
```

This reasoning can be applied to any RDF datasets that include some schema specification in RDFS.

An RDFS reasoner will be able to derive that any `foaf:Person` is also a `foaf:Agent` and that if a resource has a `foaf:familyName`, then it must be a `foaf:Person`. Particularly, it is worth noting that `rdfs:subClassOf` and `rdfs:subPropertyOf` are both transitive property, meaning that an RDF reasoner will *materialise* the complete set of `rdf:type` statements up to the *top* class. We will make extensive use of RDFS entailment in our approach. However, RDFS entailments are only one possible method to derive inferences from data. The Semantic Web community developed a set of languages to enhance RDF with inferred knowledge. For example, the W3C developed the Web Ontology Language (OWL), the Semantic Web Rule Language (SWRL), the Rule Interchange Format (RIF) and the recent Shapes Constraint Language (SHACL). Here we focus on two technologies, namely the Web Ontology Language (OWL) and the SPARQL Inferencing Notation (SPIN)[9]. The first is a W3C standard grounded in the tradition of Description Logics, having well studied computational properties and a wide range of features allowing to develop full-fledged ontologies for the Semantic Web. The second is a technology and a syntax

[9]SPIN. `http://spinrdf.org/`.

to express and execute rules using the SPARQL language syntax, initially developed by Top Quadrant [10].

The Web Ontology Language (OWL)

The motivation behind the development of OWL stands from the requirements of providing schema definitions with larger expressivity than the ones possible within RDFS. RDFS is indeed limited to subsumption relations for class and property hierarchies and the definition of properties' domain and range. The foundation of the Web Ontology Language (OWL) can be found in description logics, and therefore in first-order logic (FOL). Indeed OWL constructs are mostly based on *quantifiers*, allowing to make statements such as "any person has a name":

```
foaf:Person a owl:Class ;
  rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty foaf:name ;
    owl:someValuesFrom rdfs:Literal
  ] .
```

In what follows we introduce the basic elements of the language, taking as reference the OWL2 specification. We limit the description to the features that are used in the next chapters, the reader is referred to [W3C OWL Working Group (2012)] for further details.

OWL is a formalism to develop ontologies, and on itself, it is specified as an independent language from RDF. However, OWL has an RDF semantics that can be considered to be an extension of RDFS, as it includes RDFS entailments. For simplicity, in what follows we will use a functional-style syntax to describe OWL language constructs that is less verbose than its RDF/Turtle counterpart.

OWL ontologies include four type of statements: (a) the ontology declaration, stating that the document is an ontology and providing an identifier for it; (b) import statements, permitting to include the content of another document as an integral part of the current; (c) annotations, which are non logical statements used to document the various elements of the ontology; and (d) axioms, which constitute the logical part of the ontology, and are the ones we are going to look into some detail in what follows.

Class axioms. Class axioms include class equivalence, disjointness as well as constraints on the properties that entities of a given class can have. In OWL, entities are called *individuals*. The three

[10] Top Quadrant: `https://www.topquadrant.com/`.

sets of *classes*, *properties*, and *individuals* are mutually disjoint. Classes can be *named* (C) or anonymous, also called *class expressions* (CE).

- $SubClassOf(CE_1, CE_2)$ the subsumption relation (inherited from RDFS in the OWL/RDF semantics).
- $EquivalentClasses(CE_1...CE_n)$ the classes are equivalent, meaning any member of each one of them is also a member of the others.
- $DisjointClasses(CE_1...CE_n)$ individuals cannot belong to both classes.
- $DisjointUnion(C,\ CE_1...CE_n)$ a class is the disjoint union of a number of other classes.

Class expressions (CE) need to be part of axioms, but can be described separately, for example:

- $ObjectSomeValuesFrom(OPE, CE)$ the class of individuals having as value of the property OPE at least one individual belonging to the class defined by CE.
- $ObjectAllValuesFrom(OPE, CE)$ the class of individuals having as value of the property OPE only individuals belonging to the class defined by CE.
- $ObjectMinCardinality(n, OPE)$ the class of individuals having at least n individuals as target of the property OPE.
- $DataMaxCardinality(n, OPE)$ the class of individuals having at most n values for the property OPE.
- $DataExactCardinality(n, OPE)$ the class of individuals having exactly n values for the property OPE.

Class expressions are a useful concept as they allow to declare complex constraints by the means of anonymous classes. Here an example in RDF/Turtle syntax:

```
foaf:Person a owl:Class ;
  rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty foaf:name ;
    owl:someValuesFrom rdfs:Literal
  ] .
```

Property axioms. RDFS properties can only be defined in terms of *domain* and *range*. The main distinction is the one between object properties and data properties. OWL object properties (`owl:ObjectProperty`) have as range resources, while datatype properties (`owl:DatatypeProperty`) have literals. (A third property type is `owl:AnnotationProperty`, but they are not meant to produce inferences.) In practice, declaring a property an `owl:ObjectProperty` is equivalent to declaring its `rdfs:range` to be

`owl:Thing` (the class of all individuals). OWL includes a wide range of property features, for example the capability of referring to the inverse of an object property:

– $ObjectInverseOf(OP)$

Object properties can be of several types. In the following list, object property *expressions* OPE are meant to be a named object property or an anonymous property being the *inverse of* a named one:

– $SubObjectPropertyOf(OPE_1, OPE_2)$ the subsumption relation (inherited from RDFS in the OWL/RDF semantics).
– $EquivalentObjectProperties(OPE_1...OPE_n)$ two properties are equivalent.
– $DisjointObjectProperties(OPE_1...OPE_n)$ these relations cannot be shared between the same subject and object (for example, *parentOf*, *husband* and *childOf* are three disjoint relations).
– $InverseObjectProperties(OPE_1, OPE_2)$ the two properties are one the inverse of the other.
– $FunctionalObjectProperty(OPE)$ the object property is *functional*, meaning that for a given subject only one object is possible. The effect is that two resources being both the object of a functional property on the same subject will be interpreted as referring to the same entity. *hasBiologicalMother* is functional, as one cannot have two different biological mothers (if one have two, they must be the same person...).
– $InverseFunctionalObjectProperty(OPE)$ given a certain property object only one subject is possible. It is the case of the property *biologicalMotherOf*.
– $ReflexiveObjectProperty(OPE)$ A reflexive object property implies that any individual is connected by OPE to itself. The relation *knows* is an example of the sort.
– $IrreflexiveObjectProperty(OPE)$ On the contrary, an irreflexive object property disallows individuals to be connected to themselves by OPE. The relation *parentOf* is irreflexive.
– $SymmetricObjectProperty(OPE)$ A symmetric relation implies that if one individual is connected by OPE to another, then the second is also connected to the first by the same OPE. Being friends is such a relation.
– $AsymmetricObjectProperty(OPE)$ Asymmetry implies that the above case is impossible. Again, *parentOf* and *childOf* are both asymmetric.
– $TransitiveObjectProperty(OPE)$ Transitivity implies that if individual x is connected to y by OPE, and that y is connected to z by OPE, then x is connected to z by OPE.

Individual axioms. Also, individuals can be elements of logical axioms:

- $SameIndividual(a_1...an)$ this is the common `owl:sameAs` property used in the Linked Data.
- $DifferentIndividuals(a_1...an)$ can be used to declare that a set of individuals are all different. As a result, deriving $SameIndividual(a_1, a_2)$ would raise an inconsistency.

OWL2 includes more then what has been introduced so far, for example *punning* and *property chains*. However, we refer to the W3C specifications for a complete overview of the OWL language [W3C OWL Working Group (2012)].

A good property of the OWL language is that its design has been conducted on one hand considering its computational properties, on the other hand to allow for a rich expressivity. By doing so, the specification delivers a set of OWL profiles, having different computational properties. In fact, OWL2 profiles can be considered sub-languages (syntactic subsets) offer significant advantages in particular application scenarios.

- **OWL 2 EL** only supports a subset of the possible class restrictions (excluding, for example, cardinality restrictions) and excludes a wide range of property types (including functional and inverse functional properties). It is recommended for large knowledge bases and guarantees that instance checking, class subsumption and ontology consistency can be decided in polynomial time.
- **OWL 2 QL** guarantees sound and complete query answering with logarithmic complexity in proportion of the size of the data (assertions). This profile includes most of the features of description logic languages. It restricts the language not only in terms of constructs to be used (for example `owl:sameAs` as well as any cardinality restrictions) but also in places where some can be used.
- **OWL 2 RL** supports all axioms of OWL 2 with the exception of the disjoint union of classes and reflexivity of object properties.

SPARQL Inferencing Notation (SPIN)

SPARQL can be used to write rules as the ones described from RDFS entailments using `CONSTRUCT` queries. However, a rule engine would need to execute the query over the data iteratively until no more new triples can be derived. The SPARQL Inference Notation allows sophisticated methods to define constraints and rules using the SPARQL syntax and to *attach* them to specific elements of the dataset.

In a nutshell, SPIN:

1. Specifies an RDF vocabulary to express SPARQL queries in RDF.

2. Declares the property `spin:constraint` that can be used to link a class to a SPARQL `ASK` query that can be used to perform consistency checks.
3. Declares the property `spin:rule` that can be used to link a class to a SPARQL `CONSTRUCT` query that can be used to produce inferred triples from the members of the class.
4. Defines a `spin:Template` type useful to store parametrized SPARQL queries.

While OWL is a modelling language, SPIN is a technique to make interact queries and data in an automated way, therefore it does not have the properties of a formal language. However, it provides a large flexibility, allowing to define Horn-style rules and production rules (creating terms previously not existing in the data).

2.5 Data cataloguing on the Web

In this section we summarise the main approaches to the cataloguing of *datasets*, focusing on the way they support *terms and conditions* in the context of the Web of Data. Moreover, we give an overview of data cataloguing platforms and how they support the reference to licences and terms of use.

2.5.1 Approaches to data cataloguing

Technologies and standards for data representation and annotation have a long tradition in library cataloguing and interoperability (for example in the context of the Open Data Archive initiative[11]) and in large part, they contributed to the current standards of the Semantic Web (the Dublin Core Metadata Initiative is the most prominent case[12]).

Metadata is data that provides information about the context of an asset, supporting a set of functions associated with it. In the context of data cataloguing, metadata enables dataset discovery, understanding, integration, and maintenance [Assaf et al. (2015)].

The concept is nowadays popular and embraces a wide range of structured information about digital assets and can be characterised as (a) identification, (b) administration, (c) terms and conditions, (d) content ratings, (e) structural metadata, (f) provenance, and (h) linkage [Greenberg (2005)]. In its early stage, the Web has been often described as a *library*, although its characteristics differ under many respects. However, "something very much like traditional library services will be needed to organise, access and preserve networked information" [Lynch (1997)]. The Dublin

[11] `http://www.openarchives.org/`
[12] `http://dublincore.org/`

Core Initiative was started as a means to standardise the way objects metadata could be represented, inheriting the fundamental notions from the library domain, including authorship, attribution, and copyright [Weibel (1999)]. Most cataloguing approaches that publish dataset metadata in a machine-readable way include the capability of referring to a licence or rights statement, using mechanisms that rely upon Dublin Core (DC) properties such as `dc:license` and `dc:rights`.

The Semantic Web programme promotes approaches and techniques to publish and exchange metadata on the Web, through the definition of more specialised *vocabularies*. VoID, the "Vocabulary of Interlinked Datasets", has been proposed to describe Linked Data. VoID covers several aspects of RDF datasets, from general metadata (inheriting Dublin Core terms) to access methods, internal organisation, partition, and linking [Alexander and Hausenblas (2009)]. The RDF Data Cube Vocabulary is the W3C recommendation for the publication of multidimensional data tables, mainly in the domain of statistical data. Its scope is on providing a schema to publish data, not a method to describe datasets [Reynolds and Cyganiak (2014b)]. VOAF [Vandenbussche and Vatant (2011)] and VANN[13] [Davis (2005)] have been designed to describe vocabularies in the Linked Data cloud[14]. The Asset Description Metadata Schema (ADMS) is an RDF vocabulary proposed to be a common representation and exchange format of e-Government repositories [Shukair et al. (2013)]. All these vocabularies inherit the Dublin Core properties to express licences and terms of use. The Data Documentation Initiative has developed a metadata specification for the social and behavioural sciences. The intent is to describe data sets resulting from social science research (like survey data). An RDF version of the vocabulary has been recently developed[15] to foster the publishing into the Web of Linked Data. DDI provides reusable datasets as well as reusable metadata models and standards, including a *LicenseMandate* term to include citations to legal documents providing details about the terms of use [Vardigan et al. (2008)].

The Data Catalog Vocabulary (DCAT) is a W3C recommendation for the description of data catalogues [Erickson and Maali (2014)]. DCAT defines a dataset as a collection that is published and curated by an organisation that provides access to it in one or more formats. Examples of data catalogues using DCAT are data.gov.uk[16] and the Open Government Dataset Catalog [Erickson et al. (2011)]. The DC subschema for rights and licenses is incorporated in the *DCAT* standard of the W3C for the representation of the catalogue meta-level[17] [Erickson and Maali (2014)]. DCAT introduces a further level of concretisation via the `dcat:Distribution` class, which

[13] `http://purl.org/vocab/vann/`

[14] See the Linked Open Vocabularies activity: `http://lov.okfn.org/dataset/lov/`.

[15] `http://rdf-vocabulary.ddialliance.org/discovery.html`.

[16] `http://data.gov.uk`

[17] DCAT, `http://www.w3.org/TR/2014/REC-vocab-dcat-20140116/`

accommodates bespoke rights statements, typically using the URIs of licence descriptions. The *HyperCat* specification follows a similar notion, however, it enforces the use of URIs for values and contemplates machine-readable content as a possible form to which they dereference[18].

Even in eGovernment, where policy transparency is of the utmost importance, a fairly recent study made emerge a certain degree of heterogeneity when it comes to expressing licenses in government data catalogues [Maali et al. (2010)], although such a survey could be expected to deliver slightly more encouraging results if carried out today, if anything because of the standardisation efforts that have since been promoted by organisations like the Open Data Institute (ODI), among others [Attard et al. (2015)].

Although the importance of licence information is recognised in the scientific literature to allow businesses to be aware of the legal implications concerning (open) data reuse, also including a machine-readable representation of the licence, the existing work is still preliminary on the modelling side [Assaf et al. (2015)].

2.5.2 Tools for data cataloguing

Systems like *CKAN*, one of the best-known data cataloguing platforms, and *Dataverse*[19], support the attachment of a licence to datasets. *Socrata*[20] supports the specification of roles and permissions for the management workflow, while data terms of use are exclusively in human-readable form[21].

CKAN adopts a package manager paradigm to implement dataset management[22]. A *package*, i.e. the basic unit whereupon licences are set in CKAN, is the dataset itself. A similar argument can be made for *Dataverse*[23], which adopts a method similar to those of CKAN. A survey of existing Socrata-based open data catalogues[24] accessed through the Socrata API has brought to the surface metadata about owner descriptions and roles, permissions - mostly related to the management platform - and grant inheritance policies, all using an in-house (presumably controlled) vocabulary. Custom metadata are also used for the specification of licenses, though their instances are for the

[18] HyperCat specification, `http://www.hypercat.io/standard.html`

[19] Dataverse, `http://dataverse.org`

[20] Socrata, `http://www.socrata.com`

[21] Example at the time of writing: `https://opendata.camden.gov.uk/api/views/6ikd-ep2e.json`

[22] Comprehensive Kerbal Archive Network, `http://ckan.org`

[23] Dataverse, `http://dataverse.org`

[24] Open Data Monitor, `http://www.opendatamonitor.eu`

most part in human-readable form[25]. DKAN [26] is a datasets management system based on Drupal on with a set of cataloguing, publishing and visualisation features, whose data model is akin to the CKAN one, covering information to describe datasets, resources, groups and tags [Assaf et al. (2015)].

In recent years, data repositories and registries have been growing, spanning from data cataloguing services (Datahub[27]), data collections (Wikidata[28], Europeana[29]), to platforms that manage the collection and redistribution of data (Socrata[30]). Research Data Management include the cataloguing of the assets involved in research activities, and this area is recently developing also through a push from libraries [Cox and Pinfield (2014)]. Although [Lyon (2012)] maps potential roles of the library to a research lifecycle model, including the aspect of *Research Data licensing*, the current status of data licensing in research data management infrastructures is at the early stages (with the small exception that we will mention in Section 2.6.3). Services such as Zenodo or Figshare offer the possibility of preserving and publishing assets produced by scientific research. Zenodo encourages to share the data openly and allows a variety of licences[31]. Figshare recommends the use of Creative Commons 4.0 licences and provides guidance about a set of popular licences for open data and software[32]. An emerging category of systems is City Data Hubs, whose role is to collect data mainly from sensors networks and publish them in order to support novel IoT applications. City Data Hubs need to support developers not only in obtaining data but also in assessing the policies associated with data resulting from complex pipelines [Bischof et al. (2015); Bohli et al. (2015); d'Aquin et al. (2014)]. It is therefore for these systems to implement technologies that allow policies associated to derived datasets to be assessed.

2.5.3 Towards supporting an *exploitability* assessment

Data cataloguing systems are often designed to support discovery and indexing of data sources, leaving out an important dimension in data reuse, which is the support for *early analysis*. An important part of the early analysis is the applicability of the data sources to the use case at hand

[25]Example at the time of writing: `https://opendata.camden.gov.uk/api/views/6ikd-ep2e.json`

[26]DKAN: `http://getdkan.com/`

[27]Datahub. `https://old.datahub.io/`. Accessed: May, 2018.

[28]Wikidata. `https://www.wikidata.org`. Accessed: May, 2018.

[29]Europeana. `http://labs.europeana.eu/`. Accessed: May, 2018.

[30]Socrata. `https://www.socrata.com/`. Accessed: May, 2018.

[32]Zenodo, `http://help.zenodo.org/features/`, Accessed: May 2018.

[32]Figshare, `https://knowledge.figshare.com/articles/item/what-is-the-most-appropriate-license-fo` Accessed: May 2018.

considering the terms of use or licence associated with them. In this thesis we adopt the notion of Supply Chain Management, denoting the activities that optimise supplier-customer networks whose actors work together to improve the flow of materials and information from suppliers to end users [Handfield and Nichols (1999)]. By lifting the metaphor with data as the materials and metadata as the information, we abstract from the complexity of sub-problems like *data integration*, *metadata storage* or *policy management*. We have gathered from this abstraction and the above survey that, to the best of our knowledge, there is no end-to-end solution for exploitability assessment today.

Indeed, we record very limited support for licenses and terms and conditions management in existing data cataloguing approaches [Amorim et al. (2016); Assaf et al. (2015)]. The standard approach relies on using Dublin Core properties, which in general make no assumption as to whether their values should be machine-readable. Reasoning on policy propagation is fundamental for developing the support that an end user needs on assessing the exploitability of the data source for the task at hand. However, the heterogeneity in licence descriptions raises the issue of modelling the licence descriptions themselves in a machine-readable way. Since the early investigation carried out on using RDF to police resource access [Carminati et al. (2004)], the landscape of licence models has witnessed the contributions of several actors in digital rights. In the next sections, we, therefore, look at how policies can be represented and how processes can be represented.

2.6 Licenses and policies: representation and reasoning

Policies on the Web means different things in relation to security and privacy [Flavián and Guinalíu (2006); Kagal et al. (2003)], access and control (ACL) [Qu et al. (2004); Scott and Sharp (2002); Yuan and Tong (2005)], adaptable and context-aware systems (as a means to control the behaviour of complex systems [Tonti et al. (2003)]), and expression of legal knowledge (terms and conditions, licences) [Benjamins et al. (2005); Iannella (2001)]. In what follows we focus on the latter, considering approaches whose aim is to express and reason upon policies in the meaning of licences and *digital rights*, and limiting to the approaches designed to work with the WWW's architecture or principles.

2.6.1 Foundations

The legal domain has long been of interest to researchers in intelligent systems. Computer scientists developed theories and tools covering several research themes, spanning from the formalisation of legal concepts to the management of legal knowledge and the development of several research

branches related to automated reasoning - rule processing, case-based reasoning, and deontic logic [Gray et al. (1998)]. In knowledge engineering, legal ontologies emerged as a means to support intelligent systems in tasks such as classification, knowledge sharing and decision support. This interest has been inherited by the Semantic Web, whose vision nicely fits crucial requirements of legal intelligent systems such as the strong accent on question answering rather then document retrieval [Benjamins et al. (2005)]. Therefore, methodologies for the design of legal ontologies were developed in the context of the Semantic Web [Alexander (2009); Corcho et al. (2005); Gangemi et al. (2005)] as well as ontological frameworks like LRI-Core [Breuker et al. (2004)], the LKIF Core Ontology of Basic Legal Concepts [Hoekstra et al. (2007)] and the Core Legal Ontology based on the DOLCE foundational ontology [Gangemi et al. (2003)]. Sophisticated logic-based languages for policy representation and reasoning on the Web include KAoS, Rei and Ponder [Tonti et al. (2003)], although the context was more the one of self-regulating intelligent systems rather then the expression of licence agreements. However, all these initiatives are still to be complemented by an equivalent effort in the legal domain, with the objective of providing the appropriate legal framework under which these technologies would operate, for example a *meta-rule of law* to give foundation to the adoption of metadata in supporting rights regulation on the Web of data [Casanovas (2015); Casanovas et al. (2016b)]. In what follows we give an overview of three popular approaches to licence expressions on the Web: MPEG, CC-REL, and ODRL.

Research on rights expression languages (REL) started in the early nineties, when the eXtensible Right Markup Language (XrML) was developed by Xerox, and later on included as reference language for the expression of rights in MPEG-21 [Jamkhedkar and Heileman (2009)]. Challenges for policy languages include (a) the capability to unambiguously define the terms and conditions of a policy (policy *expression*), (b) the assurance that all parties are mutually aware of the policy and its implications (policy *transparency*), (c) the potential to detect incompatibilities between policies, (d) methods to track exceptions and obligations [Governatori and Iannella (2011)].

Policies can be represented on the Web in a machine-readable format. The heterogeneity in licence descriptions raises the issue of modelling the licence descriptions themselves in a machine-readable way. Since the early investigation carried out on using RDF to police resource access [Carminati et al. (2004)], the landscape of licence models has witnessed the contributions of several actors in digital rights. MPEG's modelling for digital rights management is divided into the Rights Expression Language (REL) and the Rights Data Dictionary (RDD) [Wang et al. (2005)]. A number of initiatives chose to apply a *semantic* approach for Digital Rights Management (DRM). Harmony project [Hunter (2001, 2003)] integrates copyright notions from the MPEG-21 RDD into a common ontology and OREL focuses on a formalising MPEG-21 RDD [Qu et al. (2004)].

MPEG REL and RDD have been also integrated into a single web ontology framework for digital rights management [García et al. (2007)]. MPEG-7 is a multimedia content description standard including a set of Description Schemes ("DS") and Descriptors ("D"), a language to express these schemes called the Description Definition Language (DDL), and an XML scheme for coding the description [Chang et al. (2001)]. COMM is an attempt to develop a Web ontology guaranteeing that the intended semantics of MPEG-7 is fully captured and formalised, also using DOLCE as modelling basis [Arndt et al. (2007)].

The Creative Commons (CC) consortium publishes guidelines for describing permissions, jurisdictions and requirements on works in general. The Creative Commons Rights Expression Language was designed to express Creative Commons licences in a machine-readable format using RDF[33]. However, its expressivity is limited to the permissions, prohibitions and duties in the scope of CC licences. Specifically for data, the Open Data Institute has proposed the *ODRS* vocabulary[34], which addresses licence compatibility and introduced the separation between data and content in the application of licenses.

The Open Digital Rights Language (ODRL)[35] is a language to support the definition, exchange and validation of policies [Iannella et al. (2015)]. Recently, the W3C Permissions & Obligations Expression Working Group[36] followed up on ODRL to develop an official W3C standard for defining permissions and obligations, published as recommendation in 2018 [Steidl et al. (2018)]. The related W3C ODRL Community Group[37] discusses use cases and requirements in support of enabling interoperability and transparent communication of policies associated with software, services, and data. The initial version was proposed for the XML language. Although ODRL is also available as an ontology, it only defines the semantics of policies in terms of natural language descriptions. An extension of the ODRL semantics has been proposed in [Steyskal and Polleres (2015)] by considering dependencies between actions, and discussing the impact of explicit and implicit dependencies on the evaluation of policy expressions.

Recently, online repositories were developed to publish licenses expressed in RDF, including *LicenseDB*[38], which uses a mostly in-house vocabulary, and the RDFLicense Dataset of the Uni-

[33]Creative Commons rights language, `https://creativecommons.org/ns`

[34]Open Data Rights Statement Vocabulary, `http://schema.theodi.org/odrs`

[35]ODRL Vocabulary & Expression, `https://www.w3.org/TR/2016/WD-vocab-odrl-20160721/`, Accessed: May, 2018.

[36]W3c Permissions & Obligations Working Group, `https://www.w3.org/2016/poe/wiki/Main_Page`, Accessed: May, 2018.

[37]W3C ODRL Community Group `https://www.w3.org/community/odrl/`, Accessed: May, 2018.

[38]LicenseDB, `http://licensedb.org`

versidad Politécnica de Madrid[39] [Rodríguez-Doncel et al. (2014)] that we use in our work. The RDFLicense Dataset is an attempt to establish a knowledge base of licence descriptions based on RDF and the ontology provided by ODRL. It also uses other vocabularies aimed to extend the list of possible actions, for instance, the Linked Data Rights[40] vocabulary [Rodriguez-Doncel et al. (2013)].

In our work, we make the reasonable assumption that the policies used to express data licences are formulated with the expressivity of ODRL, in particular, the ODRL Web Ontology [Iannella et al. (2015)] that we outline in the next section.

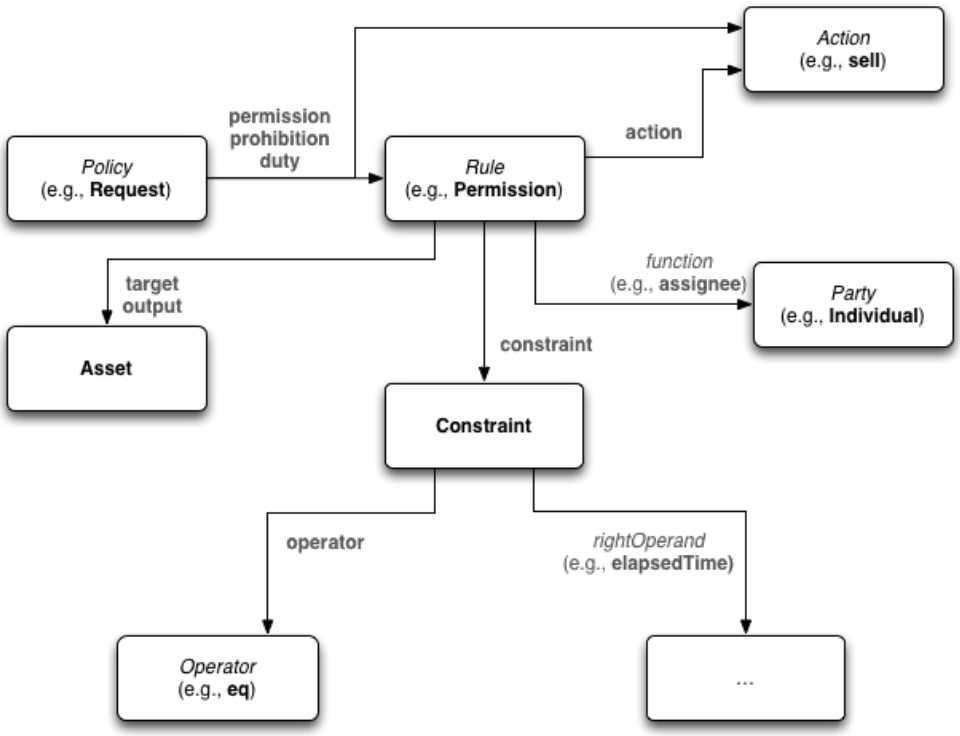


Figure 2.7: ODRL overview[41].

[39] RDFLicense Dataset, http://rdflicense.appspot.com/, Accessed: May, 2018.
[40] Linked Data Rights (LDR): http://purl.oclc.org/NET/ldr/ns#.
[41] Image taken from [Iannella et al. (2015)].

2.6.2 Open Digital Rights Language (ODRL)

The Open Digital Rights Language (ODRL), initially developed to support the management of digital rights, has evolved over the past years to a generic policy language (see Version 2.1 [Iannella et al. (2015)] and 2.2 [Steidl et al. (2018)]). Figure 2.7 shows the core entities of the underlying information model: *Asset*, *Policy*, *Action*, *Party*, *Rule*, and *Constraint*.

`odrl:Asset` is the top level entity type in ODRL, and represents any thing that can be subject to a policy (including a policy document itself). `odrl:Policy` is a generic class representing an entity meant for the expression of policies. A policy can be associated with an `odrl:Asset` with the `odrl:target` property. A policy can express an `odrl:Agreement`, `odrl:Offer`, and a `odrl:Request`. Moreover, a `odrl:Privacy` policy would refer to personal information contained in an `odrl:Asset`. Access to a given Asset can also be granted by the means of a `odrl:Ticket`, which allowance is represented as policy. In all other cases, a `odrl:Set` can be used to represent a composite policy. A `odrl:Action` can be object of a `odrl:permission`, `odrl:prohibition`, or `odrl:duty`. The language includes a vocabulary of instances of this class, including: `odrl:copy`, `odrl:user`, `odrl:derive`, `odrl:read`, `odrl:print`, `odrl:preview`, `odrl:distribute` and so forth. A policy can be associated with different parties (`odrl:Party`) depending on the role they play. For example, a party can be `odrl:assigner` or `odrl:assignee` of a policy, can be the one from which consent should be obtained - `odrl:consentingParty`, or the one to be informed - `odrl:informedParty`.

Assets can be then associated to licence documents by the means of the `odrl:target` property:

```
<http://purl.org/NET/rdflicense/OGL1.0> odrl:target <my-asset>
```

that is equivalent to using the Dublic Core term `dct:licence`: `dct:license` property:

```
<my-asset> dct:licence <http://purl.org/NET/rdflicense/OGL1.0>
```

While permissions, prohibitions and duties can be associated to single actions, more often they result from the combination of composite rules (`odrl:Permission`, `odrl:Prohibition`, or `odrl:Duty`) and constraints. ODRL can express the content of a data licence like the Open Goverment licence (OGL) described in the RDF Licences Database[42], as showed in Listing 2.3.

Listing 2.3: Open Government licence v1.0.

```
<http://purl.org/NET/rdflicense/OGL1.0> a odrl:Set;
    dct:hasVersion "1.0" ;
    rdfs:label   "UK NonCommercial Government licence v1.0";
    dct:language <http://www.lexvo.org/page/iso639-3/eng>;
    cc:legalcode <http://www.nationalarchives.gov.uk/doc/non-commercial-
       government-licence/>;
    odrl:permission [
                a odrl:Permission;
                odrl:action odrl:copy;
                odrl:action odrl:distribute;
                odrl:action odrl:derive ;
                odrl:duty [
                       a odrl:Duty;
                       odrl:action odrl:attribute;
                       odrl:action odrl:attachPolicy ]] ;
          odrl:prohibition [
                a odrl:Prohibition;
                odrl:action odrl:commercialize ] .
```

2.6.3 Reasoning with licences on the Web

Licences have an important role in regulating content policies on the Web. In this Section, we survey the problems and approaches for reasoning with licenses.

A relevant area of application relates to software licences. Free and Open Source software (FOSS) gives the possibility to reuse third-party components in the development of new systems. Many different licences have appeared including various permissions and duties on the software use: GNU General Public licence (GPL), Apache License, MIT License, to name just a few. Generally speaking, the core objective of FOSS licences is to regulate distribution. As such, they are categorised in [Kapitsaki et al. (2015)] as follows:

- (a) *permissive* - allowing software to be distributed under any licence (e.g. MIT, BSD, Apache licence 2.0);
- (b) weak copyleft - if the licensed software is modified then the resulting artefact must be distributed under the same license. Also, if the derivation does not imply modification (but simple use) then any licence can be used (e.g. GNU Lesser General Public License, Mozilla Public License);
- (c) strong copyleft - any derivative work should use the same licence (e.g. GNU General Public License, Open Software License).

Software licenses are typically assigned at component level. Normally, even small applications are composed of a large number of reused software components, and even determining all the

[42] RDF Licences Database: `https://datahub.io/es/dataset/rdflicense`.

licence involved in a given artefact is not a trivial task [German et al. (2010a); Tuunanen et al. (2009)]. Approaches are usually based on ad-hoc regular expressions on textual snippets extracted from source files or software archives [Di Penta et al. (2010); German et al. (2010b); Manabe et al. (2010)].

Apart from identification, the other two major problems in this area are licence selection and compatibility analysis. License *selection* pertains to the problem of choosing the right licence for an open source software project and is tackled by classifying the licenses. For example, formal lists of licenses are available from the Free Software Foundation (FSF) and the Open Source Initiative (OSI). However, sometimes this approach generates controversial results, as some licenses are classified differently or there is disagreement on the interpretation of the license. Indeed, in some cases, the licence classification is not straightforward.

Software reuse, repackaging and redistribution bring the problem of assessing whether the licenses associated with the respective components are compatible. A given licence X is said to be one-way compatible with licence Y if the latter can be used in association with a software resulting as a combination of components released with licence X or Y [Kapitsaki et al. (2015)]. As such, licence compatibility can be analysed by providing a graph-based model in which licenses are nodes and arcs represent the one way, non-transitive, *compatible-with* relation (like in [Foukarakis et al. (2012)]). An alternative approach attempt to model the actual content of the licence with a specific meta-model [Alspaugh et al. (2009)]. Systems for supporting the construction, visualisation and comparison of arguments have been used in order to analyse the different software licences and compare their properties [Gordon (2011)].

The Software Package Data Exchange (SPDX) specification developed in the context of the Linux Foundation[43] is promoted in order to standardise the way metadata about software components can be expressed and embedded in packaged releases, including the licenses and copyrights associated. SPDX defines also a controlled set of identifiers for licences, listing more the 300 licenses[44]. To date, licence selection in open source software repositories is proposed as a mere choice between a set of predefined licenses, without any support (for example in SourceForge[45] and GitHub[46]). From our perspective, the limit of these approaches is that they are confined to the problems of source code and software reuse and redistribution. These approaches can be of limited use in understanding the actual content of the licenses, as they are limited to the single dimension of redistribution of a derived artefact. Therefore, it is hard to assess to what extent they can be

[43]SPDX: `https://spdx.org/`

[44]SPDX Licence list: `https://spdx.org/licenses/` - accessed 21st June 2017.

[45]SourceForge: `https://sourceforge.net/`

[46]GitHub: `https://github.com/`

generalised.

In digital rights management (DRM), rule-based representation and reasoning over policies are required in order to enable secure data access and usage on the Web. Approaches can be grouped in the ones based on MPEG [Wang et al. (2005)] and the ones relying on Semantic Web technologies [Bonatti and Olmedilla (2007); Gavriloaie et al. (2004); Li et al. (2005)]. MPEG-21 has been used as a reference model for performing licence compatibility assessment and recommendation, emerging from the scenario of aggregating open and private datasets in large smart city infrastructures [Bellini et al. (2016)]. A first order logic semantics for ODRL/XML has been proposed, and used to determine precisely when a permission is implied by a set of ODRL statements, showing that answering such questions is a decidable NP-hard problem [Pucella and Weissman (2006)]. Reasoning with ODRL has been also studied by defining its semantics according to *deontic logic*. In the latter, defeasible logic is used to reason with deontic statements, for example, to check compatibility of licenses or to validate constraints attached to components on multi-agent systems [Sensoy et al. (2012)].

Compatibility and composition are two fundamental problems with licensed content in the Web of data [Governatori et al. (2013b); Rotolo et al. (2013); Villata and Gandon (2012)]. The problem of licenses' compatibility has been extensively studied in the literature and tools that can perform such assessment do exist [Governatori et al. (2014); Lam and Governatori (2009)]. Querying the Web of data implies integrating distributed data sources with a variety of policies attached to them. It is therefore problematic to assess the policies that should be associated with the query output, under the assumption that the query process is a way to combine the several data sources into a new one. This problem has been studied by applying two composition heuristics, AND-composition and OR-composition, and relying on ODRL and the rule-based reasoner SPINdle [Lam and Governatori (2009)]. The deontic components specified by the source licenses can be combined such to determine the acceptability of the operation (whether the policies of the combined data are conflicting or not) but also a new ad-hoc licence can be generated by combining the original policies [Governatori et al. (2013a)]. Associating a licence to a dataset or service is a fundamental task when publishing on the Web. A formal representation of licenses can be of use to support the users on deciding what possible constraints they want to guarantee concerning the use of their data. In [Cardellino et al. (2014)], a formalisation of common data licences in ODRL is used to implement a tool to support the selection of a data licence, and to check the compatibility of one licence with another.

2.6.4 Actions *propagating* policies: a missing link

In our work, we use ODRL as reference language for licence representation, and the RDFLicense dataset[47] as the repository of modelled policy documents. There are several tasks that can be supported by a formalised expression of policy statements. We mentioned licence identification, where users explore the different options and choose the licence that offers the best combination of permissions and constraints acceptable for the use case at hand. Combining assets is also a problematic aspect of content reuse on the Web. This problem is well known by software companies and developers as component reuse is a fundamental element of a sustainable software production process, and is becoming increasingly relevant also for what concerns data reuse. Querying distributed data sources with heterogeneous policies is clearly problematic. Once the operation is reduced to merging two (or more) datasets, the problem can be solved in terms of assessing the consistency of a policy set resulting from the combination of all the policies of the licences involved. Moreover, once this has been checked, it is also possible to propose a minimal common licence to apply to the output dataset, which would satisfy the requirements of all the original licences involved in the combination. However, the state of the art is limited to investigate the *merge* operation in data reuse, without considering the role that formalised data manipulation processes (workflows) can have on assessing the impact that original policies can have on the output. The objective of the present work is to fill the gap in the state of the art by studying the missing link between licences and workflows and to add *propagation* to the research agenda of policy reasoning in the Web of data.

2.7 Representation of process knowledge

Process modelling refers to the formal representation (a model) of events, roles and actions, developed with the purpose of analysing, prescribing or tracing a given activity. The literature around process modelling is vast and touches several research domains, spanning from business studies to manufacturing, logistics, computer science, and software engineering. For example, the description of processes is part of traditional ETL[48] and research on data integration in database systems [Lenzerini (2002); Trujillo and Luján-Mora (2003); Vassiliadis et al. (2002)]. From a different perspective, the Linked Data Value Chain [Latif et al. (2009)] is a conceptualisation targeted to the design of business models that make use of Linked Data, with the purpose of analysing the roles of providers in Linked Data applications. However, while process modelling

[47] RDFLicense Dataset: `https://datahub.io/dataset/rdflicense`

[48] `http://en.wikipedia.org/wiki/Extract,_transform,_load`

can be observed from a variety of perspectives, in our work we will look at the problem from a *knowledge representation* point of view, by giving some background on the major modelling approaches and solutions, with special attention to the way they describe the artefacts involved and their properties.

Process Modelling has been often associated with abstract mathematical models such as Petri Nets [Petri (1962)]. A *Petri Net* is a graph model where nodes can represent events or conditions, connected by arcs representing the execution flow. Originally designed to describe chemical processes, Petri Nets, also known as place/transition nets (PT), became popular models used in the description of distributed systems. Its graphical notation supports a stepwise description of the *control flow* including choice, iteration, and concurrency. The major contribution of Petri Nets was probably the switch from a sequential model for the description of automatic systems to a model capable of expressing asynchronous parallel executions [Brauer and Reisig (2009)]. Having a formal definition of their execution semantics, Petri Nets have found large application in distributed systems, resulting in a well-developed mathematical theory [Havey (2005)] as well as numerous tools supporting them [Thong and Ameedeen (2015)]. For instance, Figure 2.8 shows a Petri Net expressing an example of *Kahn process network* (KPN), a particular model of computation where a set of deterministic sequential processes are communicating through unbounded FIFO channels [Kahn and MacQueen (1976)]. Ultimately, the main focus of Petri Nets is the analysis of the process *control flow*, for example, a typical use case being the detection of deadlocks in complex workflows in manufacturing and logistics [Li et al. (2008)]. Petri Nets had a large influence in the various conceptualisations of process modelling, were the primary focus is the control flow rather than the properties of the items involved in it.

A terminology that can be encountered often in computer science is the one of *data flow*. Data flow models of computation have been developed in order to study the computational properties of complex algorithms, particularly in parallel computing. Data flow languages are *functional-style* programming languages developed from the school of thought holding that conventional "von Neumann" processors were inherently unsuitable for the exploitation of parallelism [Ackerman (1982); Johnston et al. (2004)]. For example, this is an algorithm to solve the *Hamming problem* written in Lucid[50]:

```
h
   where
```

[49] Picture by Heikki Orsila (Public Domain), via Wikimedia Commons: https://commons.wikimedia.org/wiki/File%3AKahn_process_network_as_Petri_net.svg

[50] Lucid (Programming Language): https://en.wikipedia.org/wiki/Lucid_(programming_language)

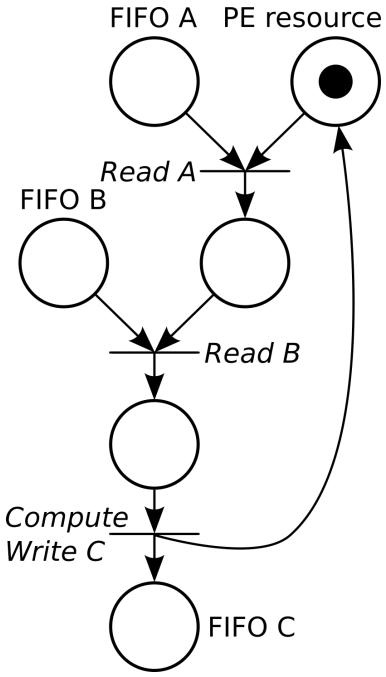


Figure 2.8: Example of Petri Net[49].

```
    h = 1 fby merge(merge(2 * h, 3 * h), 5 * h);
    merge(x,y) = if xx <= yy then xx else yy fi
       where
         xx = x upon xx <= yy;
         yy = y upon yy <= xx;
       end;
end;
```

Figure 2.9 shows a description of the related *data flow*. Similarly, a *Data Flow Diagram* is

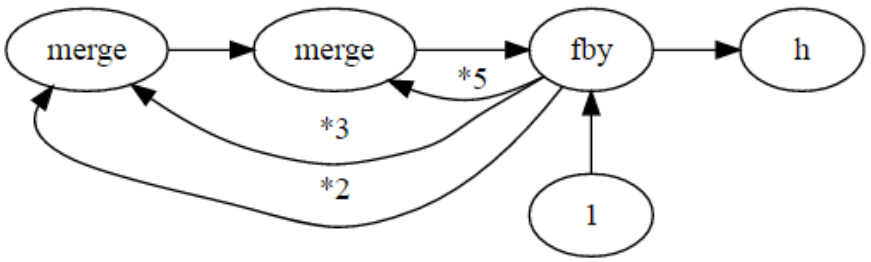


Figure 2.9: The Hamming problem designed with a data flow diagram for the Lucid programming language[51].

[51]The problem is named after Richard Hamming, who proposed the problem of finding computer algorithms for generating the ordered set of *regular* numbers (see https://oeis.org/A051037). Picture by MaksymB (Own work) [CC BY-SA 4.0 (http://creativecommons.org/licenses/by-sa/4.0)], via Wikime-

a system description technique centred on showing the *flow* of information through the various components, focusing on processing units and storage units. In this context, a data flow graph is a bipartite graph with two vertex types: *actors* and *links*. Actors describe procedures while links receive data from a single actor and send them to one or more actors [Kavi et al. (1986)]. Although the name of this technique includes a clear reference to *data*, the accent is on clarifying how these are routed through the system, independently from the type of operation performed or the effect of these actions on the content [Bruza and Van der Weide (1989)].

Process modelling is a large area, Petri Nets and data flow languages are two related approaches to the description of processes among the most influential in computer science. In what follows we focus our analysis on five thematic perspectives on process modelling. While the problem can be approached from different perspectives, the reader will see that the referred literature overlaps the different themes quite often, without being in any way exhaustive.

Business Process Modelling (BPM) is the field in which the theories and practices of process representation are developed in order to support organisations in the management of the work processes to improve efficiency, control and management of business internal and external activities [Scholz-Reiter and Stickel (2012)]. We introduce BPM from the point of view of software engineering, although the presented models received adoption in several areas within computer science (Section 2.7.1). A *knowledge level* approach to process modelling have been developed in the area of Ontology Engineering, which we survey straight after (Section 2.7.2). Research on process representation on the Web inherited most of the problems around process modelling from both software and ontology engineering. *Web Services* focused on a fine-grained semantic representation of components and data, which deserves a discussion on its own (Section 2.7.3). A *data-oriented* perspective on process modelling is the one of the research in data *provenance*, particularly important in the context of the World Wide Web. We introduce basic concepts and foundation of provenance in Section 2.7.4, focusing on the recent PROV family of specifications delivered by the W3C. Recently, Scientific Workflows have been proposed with the objective to control the life cycle of scientific experiments, but also preserve their integrity and reproducibility within the context of research data platforms (Section 2.7.5). We conclude this part by highlighting the gaps in the state of the art with relation to our research questions in Section 2.7.6.

---

dia Commons: `https://commons.wikimedia.org/wiki/File%3AHamming_problem_dataflow_diagram_(Lucid).png`

2.7.1 Process Modelling in Software Engineering

BPM research is concerned with the understanding and the management of business processes and with the design of technologies to support them. Objectives of BPM theories and models are the support of business activities through making explicit the roles and functions of the people (or the organisational units) involved, clarify the operations to be undertaken in the various phases, and standardise the actions and methods by means of clearly defined protocols. Standard bodies like OASIS[52] and The Object Management Group (OMG) played an important role in the dissemination and promotion of standards and practices of BPM in software engineering. For example, the Software and Systems Process Engineering Meta-Model (SPEM) aims at supporting the activity of software development and maintenance [OMG and Notation (2008); Ruiz-Rube et al. (2013)]. Within this context, several models have been theorised and developed with the purpose of supporting different activities such as *design*, *execution*, *monitoring*, *diagnostics*, and *interoperability* of processes [Ko et al. (2009)]. Therefore, several languages and modelling approaches have been developed, each one of them with capabilities targeted to the different purposes.

Process design is supported by visual languages and graphical tools. The Unified Modelling Language (UML) is a general-purpose family of languages popularly adopted in the field of object-oriented software engineering. UML Activity Diagrams have the purpose to support the *design* of processes (see Figure 2.10)[53]. UML Activity Diagrams (UML-AD) resemble closely the concepts of Petri Nets, including states, triggers and transitions [Havey (2005)], meant for the description of systems' processes. UML-AD defines a small set of elements focusing on the description of the control flow. Event-driven process chains (and the related XML language EPML) are also based on events and functions connected through logical operators and support sequential and parallel control flows [Mendling and Nüttgens (2006)]. The Business Process Model and Notation (BPMN)[54] provides a set of graphical elements including: Event, Activity, Gateway, Sequence flow, Message flow, Pool, Lanes and so forth. *Data objects* represent requirements or preconditions for Activities to be performed or what they produce. Data objects can represent single items or collections and can be input or output of activities. An example of BPMN model is depicted in Figure 2.11[55].

Executable process models (also called *workflows*) are developed with the objective to support software development by means of reusable components that can be combined and configured in complex chains. A workflow *engine* is a software application that executes (and manages) business

[52]OASIS: `https://www.oasis-open.org/`.

[53]Figure 2.10 was generated from `http://thoughtpad.net/alan-dean/http-headers-status.gif` (CC BY 2.5) - By Alan Dean.

[54]BPMN 2.0: `http://www.omg.org/spec/BPMN/2.0/PDF/`

[55]The example is derived from the BPMN-IO project: `https://github.com/bpmn-io/`.

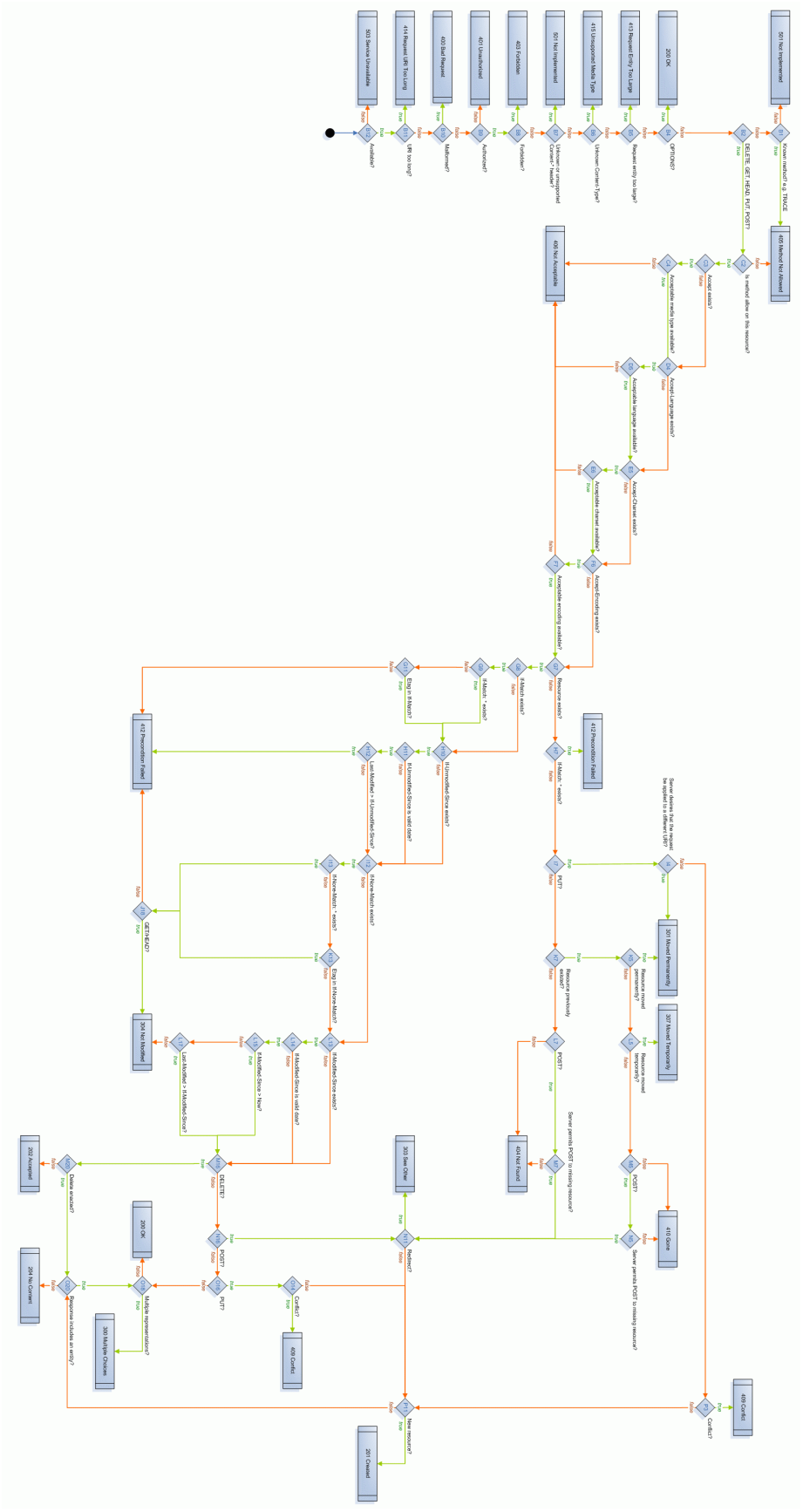


Figure 2.10: An UML activity diagram. The example describes the resolution of the HTTP response status code, given various headers.

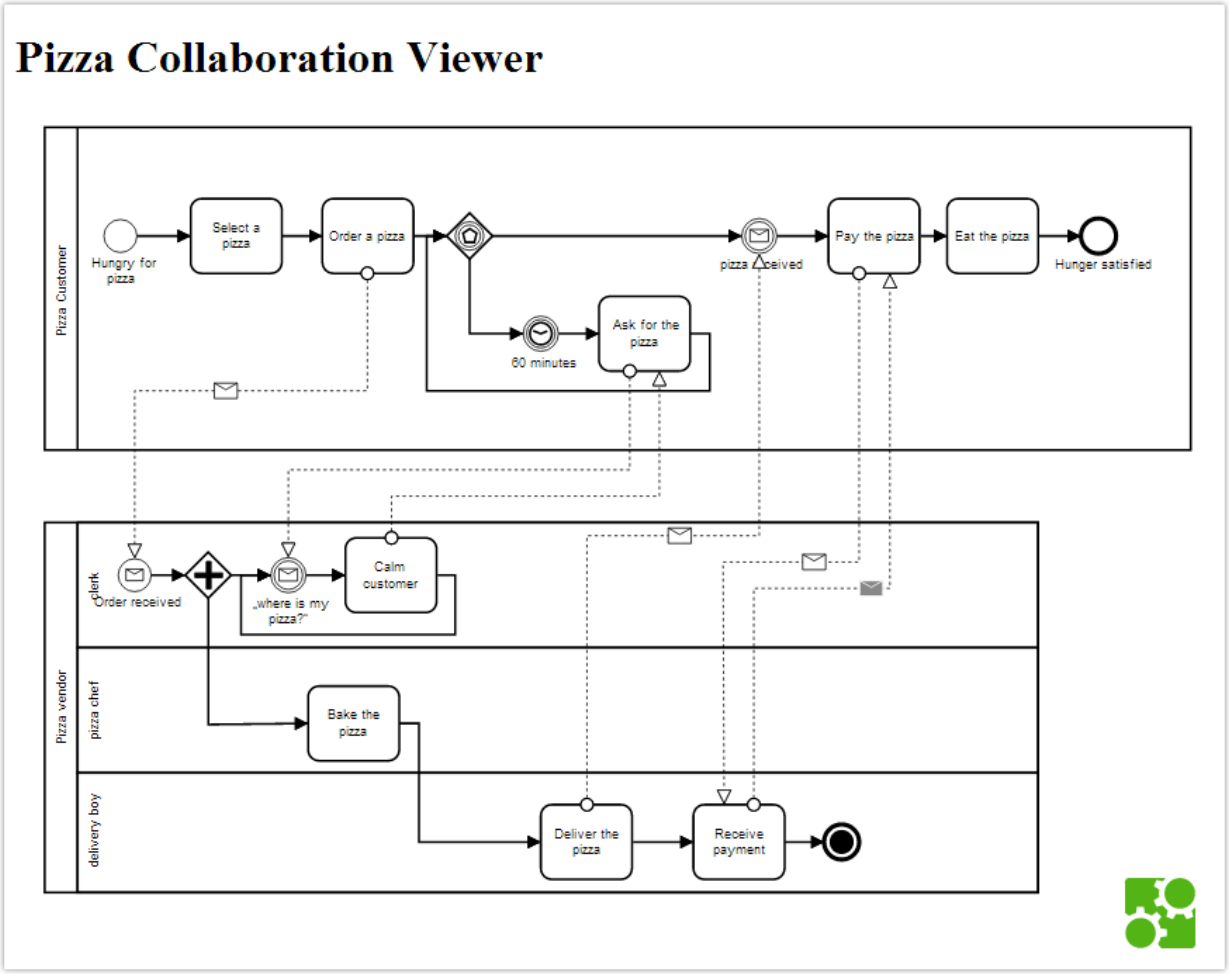


Figure 2.11: Example of BPMN model.

process models. Workflow engines can execute a given workflow multiple times by orchestrating the actions of different users (or agents) involved, allowing them to adapt the process to different data sources (configuration) and monitor its execution for quality control. OASIS[56] develops the Web Services Business Process Execution Language (WS-BPEL)[57], commonly known as BPEL, an executable language for representing actions within business processes as *Web services*. The core element of a BPEL process is an *Activity*, which specifies the nature of the operation to be performed. Some types of BPEL activities are: *receive*, *reply*, *sequence*, *if*, *while*, *repeatUntil*, *flow*. Each activity type can have a number of inputs and outputs. BPEL supports nesting of workflows (composite activities) and parallel executions. Workflow engines developed for the Business Process Execution Language (BPEL) are based on the Service Oriented Architecture (SOA) paradigm (more

[56]https://www.oasis-open.org/
[57]WS-BPEL: `http://bpel.xml.org/`

about SOA and Web Services on Section 2.7.3).

The Workflow Patterns Framework[58] was developed in the context of Process-Aware Information Systems (PAIs) with the purpose of evaluating the suitability of process modelling solutions. Workflow Patterns are a set of generic, recurring constructs under which existing modelling solutions can be evaluated [Russell et al. (2006); Wohed et al. (2006)]. Following a bottom-up approach to process modelling, this conceptual framework lead to the development of Yet Another Workflow Language (YAWL), also having its foundation on Petri Nets [van Der Aalst et al. (2003)]. Research on Workflow Patterns identified a wide range of constructs, notably the representation of a *data perspective*, delineated in four groups of patterns: *data visibility*, *data interaction*, *data transfer*, *data-based routing*. Data visibility deals with the scope of the data in the system (from local bound data objects to environmental variables). Data Interaction patterns represent the various ways in which data can be passed through the various components of the system (one to one, one to many, whether the data is pushed or pulled and so forth). Data Transfer patterns list the possible methods in which data is transferred, from shared memory to pass-by-reference or by copying, or when transformation might occur to adapt the output data to fulfil the requirements of the receiver input. Overall, Workflow Data Patterns analyse with a certain degree of accuracy the various aspects involved in the management of data in complex software processes [Russell et al. (2004)], although it has been argued that "there is no statistical underpinning" demonstrating that the patterns are representatives of real business processes [Börger (2012)].

After more than two decades of extensive research in process-aware software engineering, and the development of numerous competing standards, there is a lack of consensus about how best to describe processes [ter Hofstede et al. (2009)]. One of the main issues with traditional process modelling techniques is the trade-off between automation of model execution and intuitiveness of the language for BPM domain experts. On the one hand, easy to use tools for BPM design lack rigorous semantics, therefore the resulting models cannot effectively elucidate software requirements. On the other hand, while there are numerous workflow engines supporting more formal languages (like BPEL), these can only be used by experienced developers, therefore it is difficult for BPM domain expert to assess their capacity to fulfil the original requirements. As an example, the relation between the standard notation BPMN and its executable counterpart BPEL is insufficiently precise and incomplete, an aspect that is amply documented in the literature [Gong and Xiong (2009); Recker and Mendling (2007, 2006); Weidlich et al. (2008)]. The Subject-Oriented approach to BPM (S-BPM) is based on a controlled language that aligns BP descriptions to three basic constituents of elementary natural language expressions: *subjects* that perform *actions* on *objects*, communicating

[58] Workflow Patterns: `http://www.workflowpatterns.com/`

by *sending* or *receiving messages* [Fleischmann et al. (2014)]. Recently, a modelling practice based on Abstract State Machines (ASM) was proposed in order to provide a sound computable basis for the modelling of distributed algorithms, in an alternative to Petri Nets [Börger (2016)]. Abstract State Machines Nets (ASM Nets) can be defined by combining a textual based representation (similar to natural language) with graphical constructs (to reflect the control flow), and therefore can be combined with S-BPM in order to obtain rigorous, understandable models and potentially certifiable implementations [Börger and Fleischmann (2015)].

2.7.2 Process Modelling in Ontology Engineering

In what follows we investigate how the notion of *process* has been tackled in ontology engineering. By doing that, we omit to report how the notion has been modelled in knowledge engineering, under the assumption that the relevant approaches converged in the way ontology engineering tackled the problem. Moreover, we will omit the ontologies specifically about Semantic Web Services and Provenance, because these two themes have dedicated sections in this chapter.

As introduced in Section 2.2, an *ontology* is an "explicit specification of a conceptualisation" [Gruber (1991)]. In many cases, the term "ontology" can be associated to conceptual analysis and domain modelling, when this is accompanied by a rigorous methodology. One of the peculiarities of ontologies with respect to other types of models is the multidisciplinary approach, where logic, linguistics and cognition play a crucial role in expressing the structure of reality, often at a high degree of abstraction [Guarino et al. (1998)]. Ontologists characterised the discipline as an approach to ground logical theories on fundamental *a priori* distinctions about the nature of reality, for example, the ones between physical, objects, events, and processes [Guarino (1994)].

The Common Process Ontology (CPO) [Polyak and Tate (1998)] was developed by surveying existing standards, including the ALPS language for process specification [Catron and Ray (1991)], the Process Interchange Format (PIF) [Lee et al. (1996)], the Workow Reference Model (of the Workflow Management Coalition) [Hollingsworth and Hampshire (1995)], and the Shared Planning and Activity Representation (SPAR) [Tate (1998)][59]. The objective was to find a common ontology of processes which offer the fundamental concepts and terminology for the expression of process knowledge. CPO is modelled in three parts: a meta-ontology, an object ontology, and the constraint ontology. In CPO, a process provides a *specification of behaviour* delimited by a pair of begin/end time points, limiting the definition of behaviour as "something that one or more agents perform" [Polyak and Tate (1998)].

With few exceptions, ontology engineering is mostly concerned with developing an ontologically

[59] See also [Knutilla et al. (1998)] for a survey on process modelling of that time.

grounded definition of *Process* more than giving a tool to actually model and express concrete processes (for examples, see foundational ontologies like SUMO [Niles and Pease (2001)]). The APT framework introduces the concept of *free process* by means of five principal dimensions: *homomerity pattern*, *participant structure*, *dynamic composition*, *dynamic shape*, and *dynamic context* [Seibt (2001)].

The Process Specification Language (PSL) [Gruninger and Menzel (2003)] and ontology [Grüninger (2004)] axiomatizes a set of semantic primitives useful for describing manufacturing processes. PSL distinguishes between Activity (as class of actions), Activity-occurrence (as instance), Timepoint, and Object (anything else). A Fundamental Business Process Modelling Language (FBPML) has been proposed [Chen-Burger et al. (2002)] to fill the gap between high-level Enterprise Models (EM) and Software System Development through mediator languages such IDEF3 [Maker et al. (1992)] and PSL.

Projects like SUPER[60] produced a set of ontologies for Semantic Business Process Management (SBPM), mostly developed in WSML [De Bruijn et al. (2006)] or OCML [Motta (1998)].

DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) was developed in the context of the WonderWeb project[61] with the purpose of capturing the conceptual categories underlying human common sense through and language [Gangemi et al. (2002); Masolo et al. (2003)]. Several approaches targeted to the conceptualisation of processes originated from DOLCE. DnS (Descriptions and Situations) provides a framework for representing contexts, methods, norms, theories, situations, and models at first-order, thus allowing a partial specification of those entities. Applying a constructivist approach, it allows designing contextualised models using the types and relations defined by other foundational ontologies (or ground vocabularies) [Gangemi and Mika (2003)]. For example, in the DOLCE+DnS Plan Ontology (DDPO), DOLCE and DnS are used together to construct a Plan Ontology that includes both physical and non-physical objects, for example, events, states, regions, qualities, contextualised in *situations*. As a result, DDPO can define and regulate types of actions, their sequencing, and the controls performed on them [Gangemi et al. (2004)]. The ACL Process Ontology was developed to propose an agent-based approach to Semantic Web Services [Gibbins et al. (2004)]. Semantic Business Process Management (SBPM) is an attempt to apply ontology engineering to combine the requirements of governance in large businesses with the one of human interaction and automation [Hepp et al. (2005); Hepp and Roman (2007)]. Ontology Design Patterns (ODP) emerged to focus the development of ontologies in the Semantic Web on reusable, modular, interlinked libraries of components [Gangemi (2005); Gangemi et al. (2007)]. Many of these patterns are directly or indirectly derived from DOLCE.

[60] SUPER: `http://kmi.open.ac.uk/projects/name/SUPER`

[61] WonderWeb: `http://wonderweb.man.ac.uk/`

Extreme Design (XD) offers a methodological framework for the application of ODPs in ontology development [Presutti et al. (2009)]. Part of this effort was the development of a catalogue of *Content Patterns* (CP) [Daga et al. (2008)], generic reusable components to be used with by *specialization* (following the XD methodology), or as templates [Hammar and Presutti (2016)]. Content Patterns are often defined by combining and expanding other CPs. This is a (non-comprehensive) list of ontology patterns that can be adopted in process modelling:

- *Action.* The purpose of the pattern is to model actions that can be proposed or planned and performed or abandoned, including a specification of their status and durations in time[62].
- *AgentRole.* This pattern permits to make assertions on roles played by agents[63].
- *Role task.* Designed to connect intensional descriptions of actions (tasks) with objects (roles)[64].
- *BasicPlan* represents plans descriptions and their executions. The expansion involves the partial clone of ontology elements from DOLCE Ultra Lite and Plans Lite ontologies[65].
- *BasicPlanExecution* has the purpose of representing executions of plans along with the entity that participates in such an execution. This CP is composed of other CPs, namely *Situation* and *Region*[66].
- *Reaction* combines a number of CPs to model temporal events, for instance tracing agents and actions they produce, events that are results of some action(s), including the interpretation of dependency between actions as *reactions*[67].

The effort of the ontology engineering community on building high-quality knowledge models for the Semantic Web impacted several domains, within and outside computer science. The research mentioned above had a particular impact in the area of Semantic Web Services (SWS), which we survey in the next Section.

2.7.3 Process modelling on the Web: from Service Oriented Architectures to Semantic Web Services

Research on the automation of business processes contributed to the development of a Service Oriented Architecture (SOA) approach to system design, leading to the development of a variety of

[62] Action CP: `http://ontologydesignpatterns.org/wiki/Submissions:Action`
[63] AgentRole: `http://ontologydesignpatterns.org/wiki/Submissions:AgentRole`
[64] Role: `http://ontologydesignpatterns.org/wiki/Submissions:Role`
[65] BasicPlan: `http://ontologydesignpatterns.org/wiki/Submissions:BasicPlan`
[66] BasicPlanExecution: `http://ontologydesignpatterns.org/wiki/Submissions:BasicPlanExecution`
[67] Reaction: `http://ontologydesignpatterns.org/wiki/Submissions:Reaction`

specifications and approaches to process modelling. Although service-orientation is a technology independent approach, it is usually implemented using Web Services technologies, in particular a family of XML based specifications: the SOAP protocol for communication, WSDL for description, and UDDI for discovery and integration [Curbera et al. (2002); Fielding and Taylor (2000); Nacer and Aissani (2014)]. WSDL/SOAP applications received extensive attention from the research literature as well as implementations from all major software producers, including both proprietary platforms (Microsoft SOAP Toolkit[68], BEA Web Logic[69]) and open source ones (Apache Axis)[70]. However, their uptake as reference technology for the deployment of services on the Web has been significantly smaller than anticipated [Pedrinaci et al. (2011)], and replaced by plain Web technologies based on HTTP requests exchanging XML or JSON files. These services are called RESTful services, when they follow the REST principles [Fielding and Taylor (2000)], but more generally are referred to as Web APIs.

A variety of languages has been developed in the years to support automatic or semi-automatic Web service composition. In the XML world, the more prominent examples are WS-BPEL, WS-CDL, or ebXML (for a survey on Web service composition see [Sheng et al. (2014)]). Here we limit ourselves to the research done within the *Semantic Web*, by nature particularly interested in a semantic, *knowledge-level* description of the service, more than a functional, engineering-oriented one. Initially proposed at the beginning of the millennium [McIlraith et al. (2001)], Semantic Web Services (SWS) were envisaged as a means to enable agent-based programming capabilities for the development of intelligent systems capable to discover, compose and execute services with a high degree of autonomy. This vision initially presented as a DAML-like declarative layer to be developed as metadata of the functional services, put the basis for more than ten years of research [Nacer and Aissani (2014); Pedrinaci et al. (2011)]. Four types of semantics were identified for the description of web services:

1. *Data semantics*: the semantics about the data that is used or exposed by the service.
2. *Functional semantics*: a description of the functionalities offered by the service.
3. *Non-functional semantics*: the semantics of the aspects not directly related to the functionality, for example, the way security is implemented or characterising the reliability of the service, or pertaining the quality of the service (QoS).
4. *Execution semantics*: the description of the possible behaviours of the service, mostly related to unexpected events (runtime errors).

[68]Microsoft SOAP Toolkit: `https://msdn.microsoft.com/en-us/library/ms897379.aspx`
[69]BEA Web Logic: `https://www.oracle.com/middleware/weblogic/index.html`
[70]Apache Axis: `https://axis.apache.org/`.

Research on SWS developed around the assumption that a language covering the various semantics of the Web service would allow intelligent agents to reason upon service descriptions and support tasks including discovery and selection of services, composition, orchestration, and execution (we refer to [Pedrinaci et al. (2011)] for a complete list of tasks related to Semantic Web Services). A primary objective was therefore to develop languages and frameworks for the support of the full life-cycle of an SWS [Burstein et al. (2005); Fensel and Bussler (2002); Norton et al. (2008); Preist (2004)]. In what follows we limit ourselves to illustrate the major approaches and languages for the description of SWS (following [Pedrinaci et al. (2011)]), leaving out the variety of tasks related to web services (discovery, composition, orchestration, etc...) and the framework developed, surveyed exhaustively in [Klusch et al. (2016); Moghaddam and Davis (2014); Nacer and Aissani (2014); Rao and Su (2004); Sheng et al. (2014); Strunk (2010)]. We first introduce two top-down approaches for representing the semantics of Web services, namely OWL-S and WSMO, and then we will spend some words on a set of bottom-up approaches mostly based on a semantic annotation of existing resources (like WSDL XML files or Web pages).

Top-down approaches. OWL-S [Martin et al. (2004)] is an ontology for the description of Web Services developed with the Web Ontology Language (OWL), and evolved from DAML-S [Ankolekar et al. (2001)]. The ontology permits to represent a Web service under three fundamental dimensions:

- The *Service Profile* represents the functionality of the service in terms of input, output, preconditions and effects (IOPEs). Its objective is to advertise the service offering a high-level view of its operation. However, this is done mainly by representing its functional aspects, for example, its parameters, although the description can include a classification of the service under reference taxonomies, as well as an informal textual description.
- The *Service Model* describes the structure of the service (*how* it works), in terms of a process model expressing the semantics of a complex behaviour. This includes the sequence of operations to be performed, the type of requests, replies and the conditions under which they can occur. This level of descriptions is the one more similar to typical process modelling solutions. OWL-S identifies three process types: *atomic*, *composite*, and *simple* processes. *Atomic* processes can be directly invoked and require one single interaction to be used. *Composite* processes require several steps to be used, including multiple and conditional calls. Often, they are decomposable in other services. OWL-S supports the description of the composite process with constructs such as If-Then-Else, Sequence and Repeat-While. Simple processes are semantic abstractions over the first two types, permitting to expose their

capabilities to sophisticated reasoners, and they are not directly executable.

– The *Service Grounding* describes the way the service can be accessed, for example encoding its operational aspects in an XML/WSDL model. It includes the address of the service, the protocol to be used and the serialisation method.

The Web Services Modelling Ontology (WSMO) [Fensel et al. (2006); Roman et al. (2005)] has its roots in the Web Services modelling Framework [Fensel and Bussler (2002)], and applies an approach based on Problem-Solving methods - particularly the Unified Problem Solving Method Development Language (UPML) [Fensel et al. (2003)] - to support all the relevant aspects needed to support the discovery, selection, composition, execution and monitoring of Web services. This approach is based on a strict decoupling of the components involved and a strong focus on the role of *mediators* to orchestrate them. The four elements of WSMO are *Ontologies*, *Web Services*, *Goals* and *Mediators*, all semantically described. *Ontologies* can be defined by using *Concepts*, *Relations*, and *Functions*. Concepts are meant to establish a shared terminology to be used in the applications. Relations can define dependencies between concepts and have a binary nature. Functions can express more complex associations allowing an N-ary domain with a unary range. *Web Services* are described by means of *capabilities* and service *interfaces*. Capabilities express the functionality of the service declaring *pre-conditions* (logical constraints to be applied to the input of the service), *assumptions* (constraints about external facts that are assumed to be valid but cannot be directly checked), *post-conditions* (a set of logical constraints that express some properties of the output) and *effects* (statements about changes in the state of the world as consequence of the service execution). A *Service Interface* is a specification of how the service can be used, by means of a *choreography* and an *orchestration*. The *choreography* pertains to the client's point of view. WSMO uses Abstract State Machines (ASM) to model the interface as a *state signature* and multiple possible *transition rules*. *Orchestration* pertains to the control flow and data flow, particularly to support the use of the service in combination with others. *Goals* are used by clients to express the desired behaviour, for example in the case of service brokering. Goals are meant to express capabilities related to an expected functionality or behaviour. This notion is inherited from the notion of Task in knowledge engineering frameworks like KADS, UPML and Generic Tasks [Bylander and Chandrasekaran (1987); Fensel et al. (1999); Schreiber (2000)]. *Mediators* are defined to handle heterogeneity between two components. Mediators are expressed in terms of *source*, *target* and a pointer to the actual *mediation service*.

Bottom-up approaches. WSDL-S is an approach based on enriching WSDL Web Service descriptions with semantic annotations [Akkiraju et al. (2005)]. This approach is considered *light-weight*

(as it applies to an existing technology without requiring to adopt a specific framework) and encountered some favour because of its extendibility [Pedrinaci et al. (2011)]. In WSDL-S, it is assumed that semantic models relevant to the service already exist and are external to the service specification. In this approach, the models are maintained outside of WSDL documents, which reference them via WSDL extensibility elements. The semantics of a web service is expressed as *preconditions* and *effects*. Moreover, the specification allows the definition of mapping functions (the *schemaMapping* element) to translate data elements via XML based construct, with a role similar to WSMO mediators. Finally, WSDL-S includes a *category* element to classify the Web service under a shared taxonomy. The work on WSDL-S was adopted by the W3C working group that developed the Semantic Annotations for WSDL and XML Schema (SAWSDL) [Kopecký et al. (2007)]. The objective of the specification was to allow a semantic annotation of Web services without enforcing a specific ontological model for doing so. SAWSDL inherited the extendibility approach of WSDL-S, reducing the importance of preconditions and effects (not directly mentioned by the spec) and replacing the *category* element with a more generic *modelReference* element. Moreover, it supports referring to services to perform two operations: *lifting* - to translate some raw data to its semantic counterpart (using the *liftingSchemaMapping*), and *lowering* - to do the inverse, and extract raw data from a semantic representation (using the *loweringSchemaMapping*). However, SAWSDL does not advocate any specific language to implement (or specify) these mappings. The objective of WSMO-Lite was to exactly address this gap, providing a minimal RDFS ontology and terms to express four types of semantic annotations: *Functional semantics*, *Non-functional semantics*, *Behavioural semantics*, and *Information model* [Vitvar et al. (2008)]. Similar approaches have been developed to incorporate a semantic description of Web services in HTML pages, through Microformats (MicroWSMO [Lampe et al. (2010)]) or GRDDL and RDFa (SA-REST) [Sheth et al. (2007)]).

2.7.4 Provenance

In the Oxford English Dictionary, provenance is defined as *"the source or origin of an object; its history and pedigree; a record of the ultimate derivation and passage of an item through its various owners."* Generally speaking, provenance pertains to the origin or source of *something*, as it offers the opportunity to verify an object, assess its quality, analyse the activity that produced it, and therefore decide whether it can be trusted. The many aspects of provenance, or *lineage*, were summarised in the W7 ontological model [Ram and Liu (2009)], although most of the existing work addresses provenance as the description of data origins. In particular, provenance is often conceived as the documentation of the process [Groth (2007)]. In library studies and digital curation,

provenance pertains to the methods to assess the way an object (or a statement about an object, for example, authorship) can be attributed to a trustworthy source (although this can mean very different things depending on the context of the archive and field of application [Sweeney (2008)]). A discussion of provenance as a concept in the various research fields is in [Lemieux et al. (2016)]. Intuitively, a *provenance-aware* application is capable of answering questions regarding the origin of the data it produces, by documenting the performed processes and sources [Moreau et al. (2008)]. As a concept, provenance was developed particularly within the *database* and *e-science* communities with the purpose of tracing the *pedigree* or *lineage* of a given artefact for the sake of assessing its quality or other properties that can be derived from its origin.

In database research, provenance pertains mainly to the explanation of a query result, particularly through the collection of the *tuples* that contributed to the output of a given query. *Why-provenance* is the idea of recovering information about the so-called *witnesses* to a query execution. A witness is conceived as a subset of the database that is sufficient to guarantee that a given record is included in the output [Cui et al. (2000)]. However, *Why-provenance* does not inform on the way the output has been generated. *How-provenance* pertains the description of the derivation of an output tuple according to the query and it gives a representation on how tuples contributed to the formulation of the response, including the role of query operators (projection, aggregation, ...) [Green et al. (2007)]. A third aspect of expressing the provenance of database query results is given by *Where-provenance*, which describes the trajectory of a tuple in terms of *location*, more intuitively where a piece of data is copied from [Buneman et al. (2001)]. The *inversion* doctrine states that lineage should be computed backwards from the data that were delivered and that integration systems should provide the facilities for doing so; the *annotation* doctrine states that lineage should be pre-computed and attached as metadata to data elements [Simmhan et al. (2005a)]. Therefore, formal languages were designed for expressing the provenance of tuples, annotations and data sources with respect to the existing theory for integration rules (for example [Kondylakis et al. (2009)]). We refer the reader to [Cheney et al. (2009)] for a more detailed survey on provenance in databases.

In e-science, provenance enables the verification of scientific outputs [Simmhan et al. (2005b)]. Registers of the effects of process executions as provenance traces in scientific workflows platforms can deliver insights on the dependencies between data artefacts, support the debugging of executions, help to assess the quality of the workflow and prevent its decay [Belhajjame et al. (2013)]. *Abstract provenance graphs* have been proposed to predict data flows and dependency information occurring during scientific workflow runs *a priori*, which can be used to interpret, validate, and debug workflow results [Zinn and Ludäscher (2010)]. A template-based approach has been applied in [Garijo et al.

(2013)] with the purpose of mining common patterns in scientific workflow execution traces.

The concept of provenance is also applied to the Semantic Web, where it is essential for reasoners to make trust judgements about the information they use. Semantic Web technologies have been successfully used to express, query and reason upon provenance [Moreau (2010)]. This finds partial justification in the size of the problems of trust and provenance in Semantic Web research, which is possibly even greater than in traditional database systems: Linked Data encourage reuse of relations (as embodied in RDF properties), and rightly so, yet making it harder to trace, preserve and propagate provenance information on specific property values. In [Hartig and Zhao (2010)] the authors develop a vocabulary to annotate data with provenance information, including metadata about the process, involved software and providers. The goal is to support the selection of Linked Data sources based on trustworthiness. However, similarly to database systems research, in Semantic Web research, we record efforts on managing provenance in storage and querying, where approaches such as tSPARQL [Hartig (2009)] embed provenance data alongside actual data.

Several models have been proposed for describing process executions in the Semantic Web, like the Open Provenance Model (OPM) [Moreau et al. (2008)], the VoIDp extension of VoID [Omitola et al. (2010)], the W3C PROV Model[71] [Moreau et al. (2015)], the Provenance Model for Workflows (OPMW)[72] and more recently the Publishing Workflow Ontology (PWO)[73] introduced in [Gangemi et al. (2014)]. In particular, the Open Provenance Model (OPM) emerged as a community-driven effort for representing provenance information. OPM was the reference model for the development of the W3C PROV family of recommendations [Moreau et al. (2015)]. In what follows we focus on the W3C Recommendation *Prov-O*[74].

The PROV set of recommendations cover a variety of aspects related to provenance representation, modelling, exchange, reasoning, and querying [Missier et al. (2013); Moreau and Groth (2013)]. This family of specifications includes a conceptual data model (named the PROV model) [Moreau and Missier (2013)], an OWL Ontology [Lebo et al. (2013)], a human-readable notation [Missier and Moreau (2013)], XML serialization [Zednik et al. (2013)], a formal semantics [Cheney (2013)], a set of constraints and inference rules [Moreau et al. (2013)], and a mapping to Dublin Core [Garijo and Eckert (2013)].

The core concepts of the PROV data model are centred around the notions of *entity* - being any thing, *activity* - an action using and producing entities, and *agent* - an entity responsible for an activity being performed as it did. Figure 2.12, taken from [Moreau and Missier (2013)], illustrates

[71] W3C PROV: `https://www.w3.org/TR/prov-overview/`.
[72] OPMW: `http://www.opmw.org/`.
[73] PWO: `http://purl.org/spar/pwo`.
[74] Prov-O, `http://www.w3.org/TR/2013/REC-prov-o-20130430/`

the core concepts of PROV. Take as an example the scenario represented in Figure 2.14 - taken

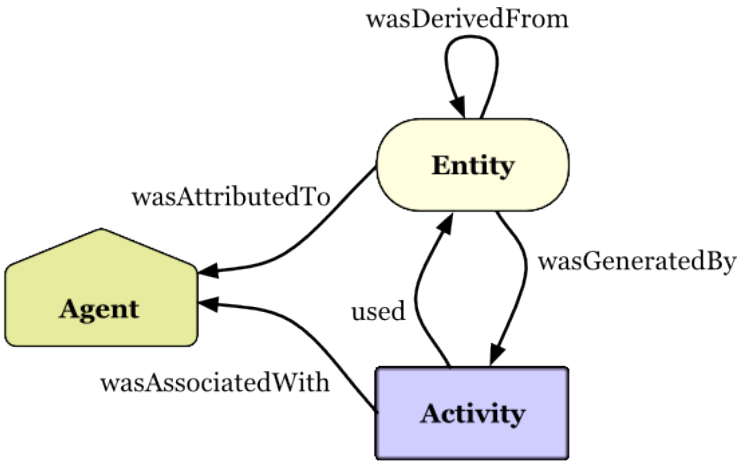


Figure 2.12: PROV core concepts.

from [Moreau et al. (2015)] - in which an online newspaper publishes an article including a chart about crime generated exploiting statistics extracted from a dataset published by a government. The article, the chart as well as the data set are all *entities*, while the process that generates the chart from the data is an *activity*. With PROV, it can be expressed that an activity *used* a dataset, and link the chart to the activity saying that the chart *was generated by* that activity, and similarly connect the chart to the dataset stating that the former *was derived from* the latter. Moreover, the dataset itself *was generated by* a publishing *activity*, therefore the previous *compile* activity *was informed by* this one. The publishing *activity was associated with* a person, who *acted on behalf of* a certain organisation. As can be seen, the PROV model can express provenance as a labelled directed graph, where nodes are activities, entities and agents, and arcs are dependencies between them.

Provenance graphs have a lot in common with workflow models. In the W3C PROV-O model the concept of *Processor* maps to the class *Activity*, in PWO with *Step*, and in OPMW to *WorkflowExecutionProcess*, just to mention some examples.

2.7.5 Scientific workflows

Research on workflows covers a variety of aspects, from the problem of reproducibility to the ones of validation, preservation, tracing and decay [Belhajjame et al. (2013); Di Francescomarino et al. (2009); Garijo and Gil (2011); Weber et al. (2008); Wolstencroft et al. (2013)]. These aspects are of primary importance in the context of science, where *"the reproducibility of scientific experiments is crucial for corroborating, consolidating and reusing new scientific discoveries"* [Garijo (2017)]. In the last decade, *Scientific Workflows* have been proposed to support the reproducibility of scientific

experiments by means of a structured description of the processing components and data artefacts involved. Particularly, these types of objects are meant to be executed *in silico*, meaning that they include all the necessary elements required to perform the experiment [Taylor et al. (2014)]. Scientific Workflows have been used in several domains, including astronomy, neuroscience or bioinformatics [Olabarriaga et al. (2014)]. Similarly to Web Services, research on Scientific Workflows involves a number of issues, spanning from their efficient execution [Callahan et al. (2006); Deelman et al. (2005); Gil et al. (2011); Ludäscher et al. (2006); Wolstencroft et al. (2013)], to workflow component reuse [De Roure et al. (2007); Goderis et al. (2005)], discovery [Wroe et al. (2007)], and recommendation [Zhang et al. (2011)]. Research Data Platforms are based on Workflow Management Systems like Pegasus [Deelman et al. (2005)], Chiron [Ogasawara et al. (2013)], Galaxy [Goecks et al. (2010)], Triana [Shields and Taylor (2004)], Kepler [Altintas et al. (2004)], Taverna [Wolstencroft et al. (2013)] (see also [Liu et al. (2015)] for a recent survey). Current research includes the problem of monitoring workflow executions for resources consumption and data quality in large research data infrastructure [Mannocci (2017)]. Recently a number of repositories of scientific workflows have been published - Wings[75], My experiment[76], SHIWA[77] are the prominent examples. From the point of view of knowledge representation, Scientific Workflows inherit the approach developed in three decades of process modelling and are structured as a directed graph of nodes as processing units linked by arcs representing a dependency relation (often as an output-to-input connection). Workflows are built on the concept of *processor* as the unit of operation[78]. A *processor* includes one or more input and output *ports*, and a specification of the operation to be performed. Processors are then linked to each other through a set of data links connecting an output port to the input of another processor resulting in a composite tree-like structure. Figure 5.10 shows an example of a workflow taken from the "My Experiment" repository[79]. A major challenge in understanding workflows is their complexity. A workflow may contain several phases, whose role in the scientific analysis can be opaque if only looking at the workflow implementation. This difficulty in understanding the intention behind implementations stands in the way of workflow components reuse. Semantic technologies have been used to analyse the components of workflows, for example, to extract common structural patterns [Ferreira et al.

[75]Wings: `\T1\textemdashhttp://www.wings-workflows.org/`.

[76]My experiment: `http://www.myexperiment.org/`.

[77]SHIWA: `http://www.shiwa-workflow.eu/wiki/-/wiki/Main/SHIWA+Repository`

[78]Here we use the terminology of the SCUFL2 specification developed in the context of the Taverna workflow management system. However, the basic structure is a common one. In Kepler, for example, this concept maps to the one of *Actor*.

[79]"LipidMaps Query" workflow from My experiment: `http://www.myexperiment.org/workflows/1052.html`.

```
Data-Operation motifs
 Data preparation
  Combine
  Filter
  Format transformation
  Input augmentation
  Output extraction
  Group
  Sort
  Split
 Data analysis
 Data cleaning
 Data movement
 Data retrieval
 Data visualization
Workflow-Oriented motifs
 Inter workflow motifs
  Atomic workflows
  Composite workflows
  Workflow overloading
 Intra workflow motifs
  Internal macros
  Human interactions
  Stateful (asynchronous) invocations
```

Figure 2.13: Workflow motifs.

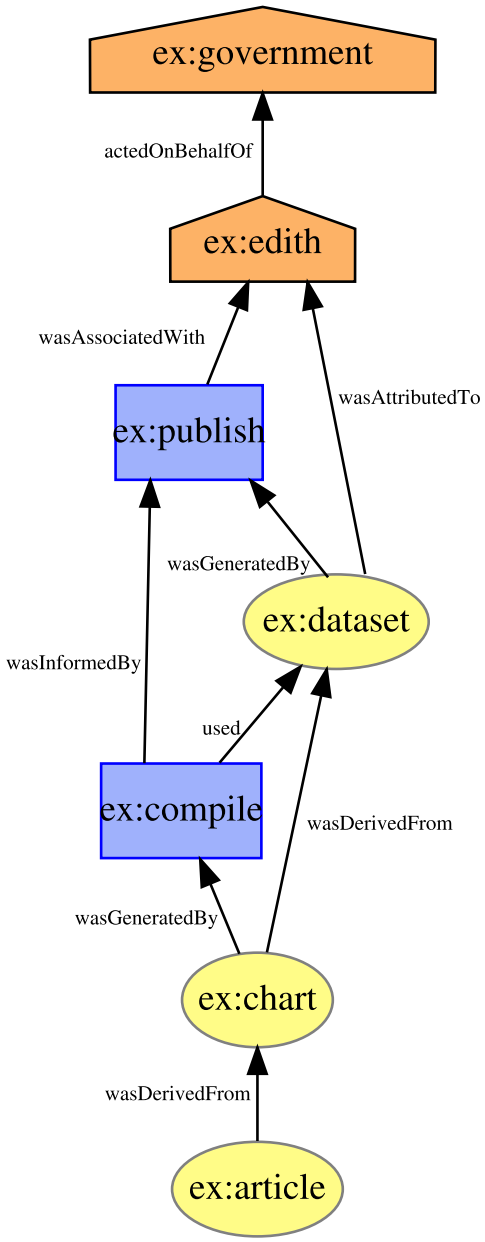


Figure 2.14: PROV example scenario. Figure taken from [Moreau et al. (2015)].

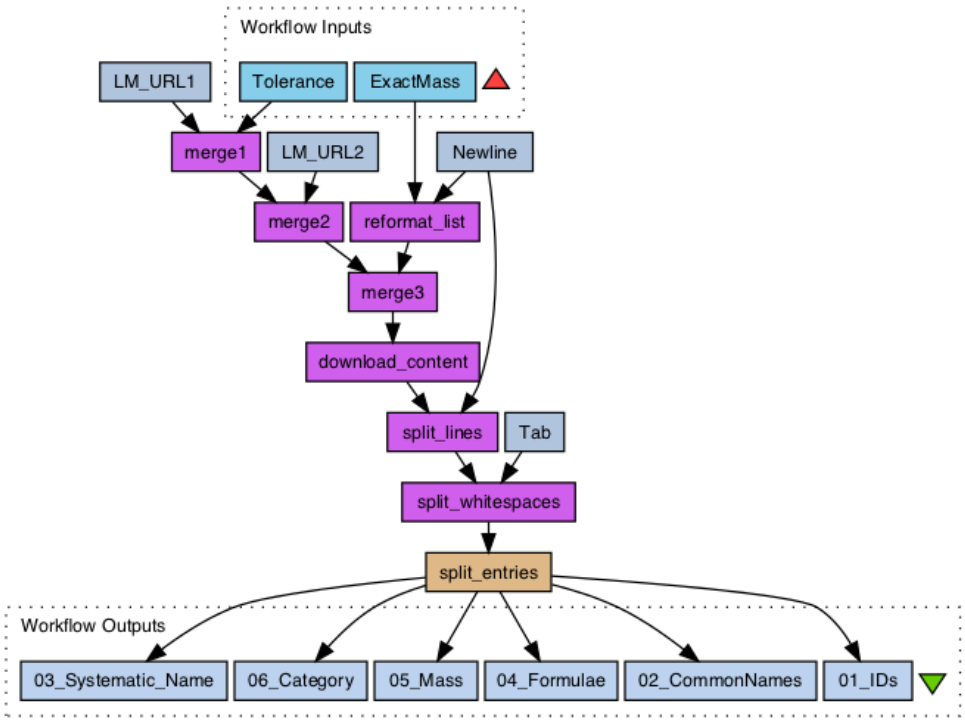
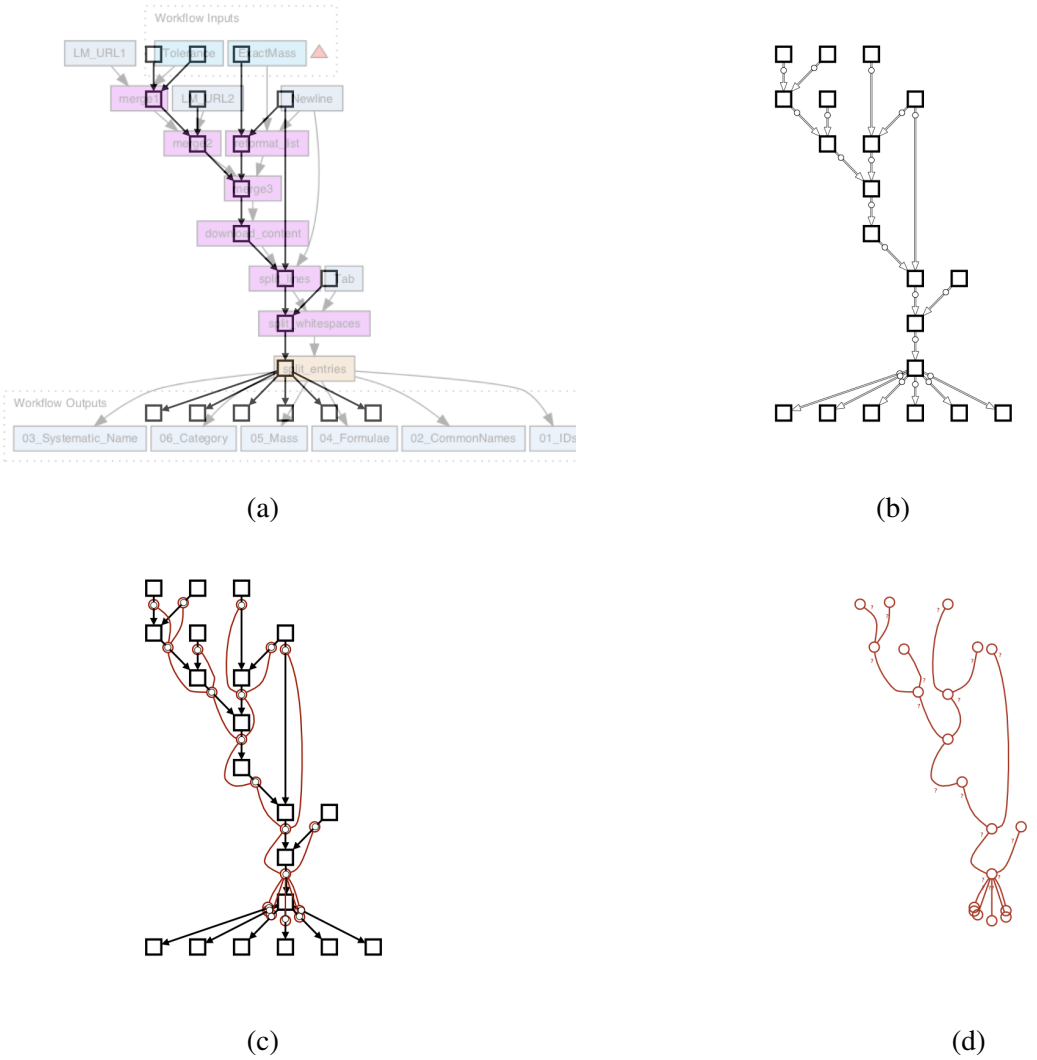


Figure 2.15: A workflow from the My Experiment repository: "LipidMaps Query".

(2011)]. Recently, more attention has been given to the elicitation of the activity of workflows in a knowledge-principled way, for example labelling data artefacts to produce high-level execution traces (provenance). This research highlighted the need for adding semantics to the representation of workflows and the challenges associated with the problem of producing such annotations [Alper et al. (2014)]. A recent line of research is focused on understanding the activities behind processes in scientific workflows, with the primary objective to support preservation and reusability of workflow components [Garijo et al. (2014)]. This analysis resulted in a set of *workflow motifs* that identified *data-intensive motifs* representing the data-relying activities that are observable in workflows, and *workflow-oriented motifs* showing the different ways in which activities are implemented. Scientific workflow motifs are shown in Figure 2.13 (content from [Garijo et al. (2014)]). An interesting discovery of this analysis is that a significant amount of data-intensive operations are related to data preparation [Garijo et al. (2014)].

However, workflow knowledge encompasses several aspects (control, data, implementation compliance, semantics), therefore it is naturally spread among several different artefacts (configurations, datasets, logs, and so forth). Workflow-centric Research Objects have been designed with the objective of making persistent (and reproducible) research experiments in the scientific discourse. The approach is to bundle all the elements relevant to a research finding in a single



(a) (b)

(c) (d)

Figure 2.16: From workflows to *dataflows*. Workflow models are focused on actions, to support multiple and parametric executions (2.16a). There are scenarios in which we need to focus on the data (2.16b) and understand how the data is affected by the actions of the workflow (2.16c). In our work, data flows 2.16d are characterised as an expression of the implications of the process on the data.

artefact, including the workflow formalisation, the required input data, the provenance of the results obtained by its enactment, and any other digital object involved (papers, datasets, etc.), as well as semantic annotations that describe all these objects [Corcho et al. (2012)].

2.7.6 Conclusions: towards a data-centric model of processes

The previous sections provided a retrospective of the fundamental research in process modelling. However, in the present thesis, we study the problem of policy propagation in data flows. Our research questions focus on the knowledge components necessary to reason on policy propagation ($R.Q.\ 1$), with the hypothesis that these components should establish a semantic relation between the process and the policies involved. The process representation should be capable of expressing the relations between the data artefacts involved. Therefore, our goal is to devise what are the possible relations between data objects involved in complex processes. A natural approach is to extend the inference capabilities of existing workflow models. Business process modelling, web services, scientific workflow, and provenance are four areas in which the problem of modelling processes have been extensively studied. Workflows are made of agents, activities, artefacts and links between them. However, process models do not take responsibility for managing the life-cycle of data properties as part of their conceptual model. Moreover, the surveyed models are successful in describing the relations between the artefacts in terms of *dependency*, but none of them focuses on qualifying further this dependency, and establishing in detail the semantic relation between the data objects involved. These relations would make it possible to infer what properties of a certain artefact are transferable to another within a given process.

The focus of the data flow models is on controlling the information flow, with no support with regard to representing the way in which the properties of the data artefacts involved can be affected by the activity itself.

The Workflow Patterns approach to process modelling in software engineering identified a *Data perspective* through which process elements can be abstracted. However, the resulting patterns do not focus on the possible semantic relations between data and processes, particularly do not help on qualifying the relation between two data items involved in a workflow.

Ontology engineering explored the concept of *process* extensively. Foundational ontologies focused on the nature of the concept more than on its characterisation in terms of reusable models. Ontology design patterns include typical elements of process models like agents, activities and roles, which can be of use in analysing how different agents participate in the overall activity, more than expressing the properties of the artefacts involved, and how they are affected by process actions.

OWL-S supports reference taxonomies for the classification of Web services. WSMO supports

Preconditions and Effects in order to establish constraints and rules for reasoning with a process model. That could be a suitable engineering solution and could help supporting reasoning on the meta properties of the data items, particularly the way policies are affected by the actions. SAWSDL follows the tradition by only offering a *method* to link services to semantic descriptions, without going into the details of the actual semantics of services, and how they affect the data items involved. WSMO-Lite, having the purpose of specifying an RDFS ontology to express the various semantics of services, still limits itself to providing *classes* of service features, instead of the actual possible features. However, these solutions do not provide support on qualifying the nature of the activities performed, and therefore the possible effects on the properties of the data.

Provenance models can express dependency between workflow participants (agents, activities, items), but this stops at the concept of *derivation*. However, different operations can have a different impact on how a policy can propagate. It is still an open question whether dependency and derivation are sufficient concepts to reason on policy propagation.

Common Motifs is a rightful attempt to semantically classify workflow processor. Such classification highlighted the possible operations that are performed in a scientific workflow. The characterisation of activities can be of use in assessing how they affect the data, although in order to be of use this needs to be expressed as relations between the data artefacts involved (see Figure 2.16).

In what follows, we investigate how policy propagation can be supported by a model that expresses the semantic relations between the artefacts involved, therefore deriving a generic reference metamodel, where a data object is related to another one. Expressing the qualities of this relation is one of the contributions of this work and the objective of the next chapter.

Chapter 3

Semantic Representation of Data Flows

The problem of propagating policies is a specific case of metadata propagation. The question is whether any of the policies derivable from the licence associated with the origin data, requires to also be part of the metadata of the target or of any of the intermediate results. However, for this question to be answered, it is necessary for the two data artefacts to be somehow related. Depending on the quality of this relation, the policy will propagate, or not. In this work, we conjecture that the relation between policies and processes can be expressed by focusing on the data items, and assume that Web systems can be described from the point of view of interlinked data objects. Therefore, in this Chapter we focus the representation of the data and the possible ways they can be associated to each other, and develop an ontology to elicit the possible *data-to-data* relations. In order to build this *knowledge component* we take Semantic Web applications as prototypical Web systems that can shed a light on the possible ways data can be manipulated on the Web of data.

In this chapter we describe an ontology, which generalises the relations between these "data objects", which we call *datanodes*. By looking at Semantic Web applications, we realised that many interesting relations can be devised from data flows of applications that, along with the ones expressed by existing vocabularies, can support the analysis, understanding, development and maintenance of data hubs. We analysed the description of the data flows of semantic web applications to observe the rationales behind these data flows. We observed that these systems have a strong focus on data nodes (datasets, ontologies, catalogues, etc.) that may emerge as input or output at different steps of the process and/or layers of an application's architecture. These variety of relations can be understood as a graph connecting nodes of data, as a means to specifying the possible relations between them. As a result, a foundational pattern emerged, where complex structures are collapsed into a set of binary relations. The resulting representation is a direct graph where the nodes are the data and the links their relations. We named the single class *Datanode*. Our

abstraction represents *a Datanode related with a Datanode* in six fundamental ways, as shown in Figure 3.1.

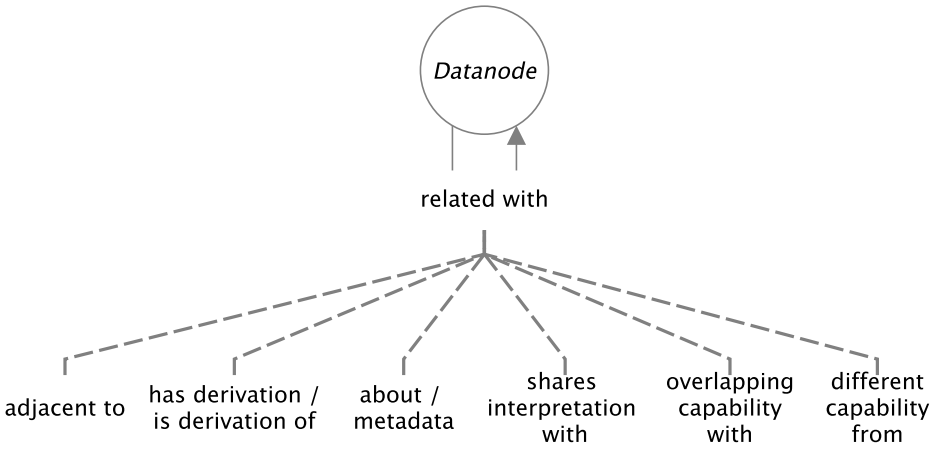


Figure 3.1: A summary of the Datanode ontology.

The chapter is structured as follows. The next section is dedicated to describing the genesis of the Datanode ontology. In Section 3.2 we describe the Ontology in detail, and we organise the hierarchy of relations that specialise it in the Datanode ontology. In this section we also report on alignments with Semantic Web ontologies as well as on basic reasoning properties of the ontology. In Section 3.3 we detail the testing of the ontology, applied to describe the set of applications used as reference sources, plus two more only used in the testing phase. In Section 3.4 we describe an exemplary scenario of Data Hub management, where we believe the Datanode approach is desirable to enhance existing solutions. We show how the Datanode approach can provide a method to reason upon some common properties of complex data flows. In Section 3.5 we discuss related vocabularies and modelling practices, in particular the relation between the Datanode ontology and existing Semantic Web ontologies. Finally, Section 3.6 reports on other possible applications of the ontology and discusses future work.

3.1 Genesis of Datanode

The genesis of Datanode is grounded on the tradition of ontology engineering [Corcho et al. (2003); Suárez-Figueroa et al. (2012); Sure et al. (2009, 2006)]. Although with several variations, these methodologies are based on the assumption that knowledge need to be acquired by human experts, for example through interviews and focus groups whose objective is the production of a ontology requirements specification document (ORSD). Such document incorporates a set of Competency Questions (QS) [Grüninger and Fox (1995); Uschold and Gruninger (1996)], a means for validating

the model produced. After requirements have been specified, the following step is essentially a construction phase, where concepts and axioms are designed, evaluated, and refined, often with a test-driven and iterative approach. The method used to develop Datanode follows these principles. However, instead of selecting human experts, we chose to follow a more empirical approach and start from the identification of representative artefacts. In particular, we look at the Web of Data to collect exemplary systems to be used as starting point for developing the requirements of our ontology. Overall, our design methodology is iterative, incremental and test-driven, and is strongly inspired by Extreme Design in ontology engineering [Presutti et al. (2012)].

Semantic Web Applications can be considered as systems operating in the World Wide Web that follow principles or incorporate technologies such as RDF and other standards [Oren (2008)]. For our purpose, that is the one of describing data manipulation processes in the Web of Data, we restrict the above definition by considering as Semantic Web Applications systems that, in addition, have all the following characteristics:

- Make use of information from sources they do not necessarily control;
- Manipulate it to produce new knowledge;
- Process the resulting data for presentation in a new form.

Our ontology design process can be summarized as follows:

1. **Selection**. Select applications described in recent contributions to Semantic Web conferences by using the definition given above.
2. **Acquisition**. Isolate the descriptions of each application from the related paper.
3. **Use case abstraction**. Generalise the description by translating it to a generic synopsis.
4. **Design**. Annotate the synopsis with names of classes and properties that appear in the text; generate a draft ontology and then refine it through discussions.
5. **Testing**. Model the synopsis using the ontology. Testing was done in parallel with the design phase, until all cases were fully covered.

In the following sections we report on the process that led to the Datanode ontology. As illustration, we will follow the evolution of a snippet extracted from one of the systems analysed, namely DBRec [Passant (2010)]. DBRec is a music recommendation system built using DBPedia. The paper [Passant (2010)], among other aspects, describes the preparation of the data done to maximise the efficiency of the Linked Data Semantic Distance algorithm.

3.1.1 Selection & acquisition

The initial phase was dedicated to the collection of exemplary applications of the Web of Data. We reviewed the content of major Semantic Web conferences (e.g. International Semantic Web Conference, Extended Semantic Web Conference, and Web Semantics) and co-located workshops between 2010 and 2014 (when the work has been conducted) and applied the restricted definition of Semantic Web application mentioned above as selection criteria. We selected six applications that fit well our definition of Semantic Web Application: Aemoo [Musetti et al. (2012)], DBRec [Passant (2010)], DiscOU [d'Aquin et al. (2012)], Spud [Kotoulas et al. (2014)], Yokohama Art Spot [Matsumura et al. (2012)], EventMedia [Khrouf and Troncy (2012)]. All these are applications that do not start from a dedicated dataset, but transform some existing sources to achieve a target task. We added Rexplore [Osborne et al. (2013)] and IBM Watson [High (2012)] at a later stage, to perform our evaluation (see Section 3.3). Although we did not perform a systematic analysis of all contributions produced in this period (there might be other significant ones), the selected cases have all the properties we want to observe: 1) the data is gathered from external sources 2) it is processed to produce new knowledge, and 3) the resulting data is offered for consumption by humans or other systems. Moreover, these articles all dedicate a significant effort on describing the architecture and operations of the system. We don't address the problem of completeness at this stage, in the absence of a general framework with a comprehensive set of features that are supposed to be included. However, we will discuss the problem of the completeness of the ontology and of other knowledge components later in this work.

We started by isolating the portions of text describing the system. When the system was including several different features and descriptions spread over many points in the document, we isolated all of the snippets and merged them in a unique picture at a later stage. This example is extracted from the discussion of a data preparation process in DBRec:

> ''On the other hand, we identified lots of redundancy and inconsistencies in our DBpedia subset. Especially, many links between resources are defined redundantly as http://dbpedia.org/ontology/ xxx and at the same time as http://dbpedia.org/pro−perty/xxx. We then removed duplicates, leading to 1,675,711 triples, i.e. only 55.7% of the original dataset.''

3.1.2 Use case abstraction

The aim of this activity was to try and extract generic use cases from specific ones. We went through each description and translated it into a list of atomic sentences, each describing a step of the process (or a specific feature). We obtained a synopsis from each text snippet. During this process we tried

to abstract from the specific examples, replacing peculiar aspects with their generalisation, when possible. We obtained a set of synopses, covering several aspects of Semantic Web Applications, similar to the following:

1. having data with redundancies and inconsistencies
2. remove redundancies and remove inconsistencies
3. remove duplicates (same as redundancies?)
4. data is now 55.7 percent of before

As a result of this process we obtained a list of text snippets describing system features, each coupled with a synopsis. The whole set of synopses constituted our modelling requirements[1].

3.1.3 Ontology design

We traversed the collection of synopses and treated each sentence on its own. We annotated each sentence with a set of keywords. We then looked at the keywords and modified them in order to make them represent either a verb or a type of things. In the second case, we wrote it capitalised:

0) access, Data, Redundancy, Inconsistency
1) remove, Redundancy, Inconsistency
2) remove, Duplicate
3) Data, Now, Before

We traversed all synopses several times, trying to harmonise the concepts (e.g., merging synonyms) while maintaining the specificities emerging from each case.

We then collected all 159 keywords from our annotations, analysed them trying to reduce redundancy (for example normalising names to be singular). The result was a list of names of possible concepts and relations. We automatically generated a draft ontology considering all concepts to be instances of `owl:Class` and all relations to be object properties between (yet) unspecified things. This draft ontology included 132 classes and 168 properties, covering several different aspects of a Semantic Web Application. Here is a sample list: Operation, Collect, Optimize, Capability, newer-version-of, not-fits, before, Entity, Dataset, about, Summarize, component-of, Stand-in, concern-of, derive-from. This phase required to abstract and simplify this automatic, disorganised and heterogeneous ontology. This was based on three main observations:

1. Many of the types represented data sources or objects in various forms;

[1] We do not explicitely formulated these requirements as Competency Questions (CQ) [Grüninger and Fox (1995)]. Nevertheless, they had the role of competency questions in our methodology, guiding the design and testing of the ontology.

2. Many of the processes could be represented through the relationships between their input data and their output data.
3. We can organise the relationships hierarchically, and infer abstract notions from specific relations

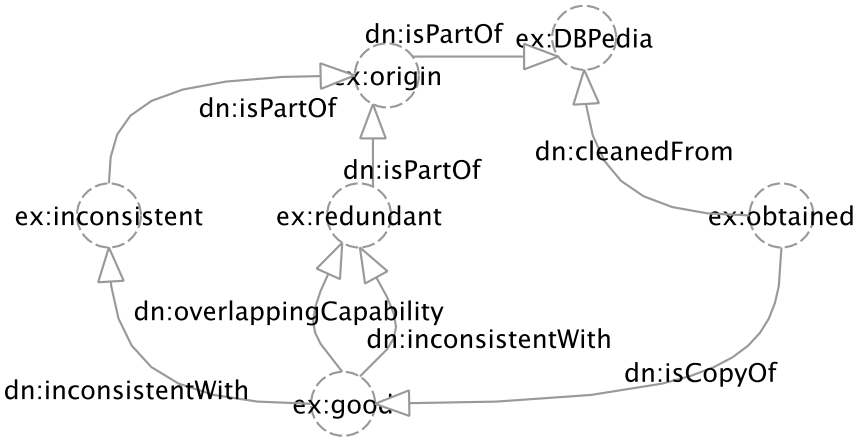


Figure 3.2: Example test case, from DBRec [Passant (2010)]. The graph shows how the process that goes from the DBPedia dataset to an optimised and curated dataset can be formalised with Datanode. The datanode *ex:good* is the result of the whole process, it is described as a copy of the datanode obtained as a result of a data cleaning of DBPedia targeted to the task (right side of the picture). In other terms, only a subset of DBPedia has beed considered - *ex:origin*, the corrupted part *ex:inconsistent* has been left out (is not compatible with the result: *dn:inconsistentWith*), and that the result has overlapping capabilities with a *ex:redundant* part that has been also left out from the result (the centre of the picture). We will detail these and other relations in Section 3.2.

With this foundation established, we then refactored all the elements obtained in the previous steps so that they are either abstracted into the class Datanode, or modelled as properties of the class Datanode. We formalised each relation and organised them in a hierarchy, testing the changes with the available use cases iteratively. Figure 3.2 shows one of these test cases. We refined the property hierarchy specifying ontological characteristics - symmetricity, transitivity, etc. - when appropriate. We grouped all relations under the top property `relatedWith` (we will discuss the pattern in depth in Section 3.2).

3.1.4 Testing

We iteratively modelled all synopses with the hierarchy of relations, and verified that the inferred model was fitting the intended meaning of the description. For testing the ontology we considered

two more applications, namely Rexplore [Osborne et al. (2013)] and IBM Watson [High (2012)], to strengthen our evaluation. On each test iteration we discovered new implications derived from the organisation of the properties as a composite hierarchy (in the spirit of the Extreme Design [Presutti et al. (2009)] methodology).

3.1.5 Result

Datanode is an ontology constructed by specialising an essential pattern - a Datanode related with another - in the spirit of Content Ontology Patterns [Presutti et al. (2009)], whose domain of application is data catalogue description and management.

The foundation of the Datanode Ontology is therefore devised as a simple abstraction including a unique class, Datanode, which represents "data" in a broad sense (data sources of any kind or format), and formalised relationships between these datanodes. The top layer of the property hierarchy shows six fundamental aspects through which datanodes are related: metalevel, derivation, overlapping capability, different capability, shared interpretation and adjacency (see also Figure 3.1). These are described in detail in the following section.

3.2 The Datanode Ontology

The Datanode ontology is based on a fundamental design choice. We found data to have several different facets in the systems' descriptions, examples include: sets of data, data sources, identifiers, metalevel descriptions. The Datanode class abstracts from the notion of dataset and it is independent from the notions of containment, individual or identifier. A *datanode* is any data artefact. The class groups datasets, ontologies, schema elements such as classes and properties, as well as identifiers under the same umbrella. The nature of this concept is voluntarily underspecified. The ontology defines a unique type - `Datanode` - and six top relations, starting from a single top property: `relatedWith`, having Datanode as domain and range, as shown in Figure 3.1. The Datanode terms are under the following namespace:

`http://purl.com/datanode/ns/`

The Datanode top hierarchy is specialised by extending the six "branches", thus expressing more specific relations. Table 3.1 lists a number of abstract competency questions that illustrate the rationale behind the structure of the ontology. In the following sections we describe the top relations and the specialised relations that compose the Datanode Ontology [2]. Tables will list the relations,

[2]Documentation is online at `http://purl.org/datanode/docs/`.

Table 3.1: Competency questions mapped to abstract relations in Datanode. Each CQ should be read considering a given Datanode as focus.

Aspect	**CQ: Given a Datanode...**
Metalevels	Which are the datanodes that contain metadata about it? Which datanodes are described in it?
Derivation	Which datanodes has been produced by activities that used it as input? Is it the result of a manipulation, observation or processing of another datanode?
Adjacency	Which node shares the same container? Which nodes are part of the same catalogue? Which datanodes are part of the same datanode?
Capabilities (overlap)	Which node shares a feature with it?
Capabilities (different)	Which node has different features?
Shared interpretation	Which datanode can have a role on affecting the inferences that can be derived?

their OWL type and relations they are `rdfs:subPropertyOf`. Table 3.2 lists the symbols used to indicate OWL property types.

3.2.1 Metalevels

This branch covers the relations between something and its metadata. The property `metadata` is used to designate a relation with information that applies to the datanode as a whole, possibly including it. This relation has for inverse `about`.

This kind of relation specialises as `describes` / `describedBy`, `hasAnnotation` / `isAnnotationOf` and `hasStatistic` / `isStatisticOf` (we will follow this convention to pair relations that are inverse of each other).

Table 3.3 shows the structure of this branch[3].

[3]The reader will note that many relations fall under more then one branch (for example, the relation `hasStatistic` is a specialisation of `describedBy` as well as of `hasComputation`).

Table 3.2: Legend of OWL 2 Property types used in Tables 3.3-3.8

Symbol	**OWL Property type**
T	TransitiveProperty
S	SymmetricProperty
R	ReflexiveProperty
F	FunctionalProperty
I	InverseFunctionalProperty
X	IrreflexiveProperty

Table 3.3: Properties in the Metalevels branch

relation	**sub property of**
metadata	relatedWith
about	relatedWith
describedBy	metadata
describes	about
isAnnotationOf	about, attachedTo
hasAnnotation	hasAttached, metadata
hasStatistic	describedBy, hasComputation
isStatisticOf	describes, isComputationOf

3.2.2 Derivations

This relation indicates that a datanode is the origin of another, in the sense that the second has been produced using the first as information source. It is possible that a source `hasInterpretation` / `isInterpretationOf` some other data, as a result of a mining process - `hasExtraction` / `isExtractionOf`; or a logical process - `hasInference` / `isInferenceOf`. A relation `processedFrom` / `processedInto` is identified when a derivation is also the transformation of a data node into another one, through being cleaned, optimised, refactored, remodelled or the result of a computation - `hasComputation` / `isComputationOf`. It is possible to derive a new data node by making a copy - `hasCopy` / `isCopyOf`, when making snapshots or caching data (`hasSnapshot` / `isSnapshotOf`, `hasCache` / `isCacheOf`). We consider also the case when a derivation is done by cloning a part of a datanode with specific features - `hasSelection` / `isSelectionOf`.

Table 3.4 shows the full list of relations that are part of this branch.

Table 3.4: Properties in the derivation branch

relation	**sub property of**
isDerivationOf	relatedWith
hasDerivation	relatedWith
isSummarizationOf	isDerivationOf
hasStandIn	hasDerivation, overlappingCapabilityWith
processedInto	hasDerivation
combinedIn	hasDerivation
hasSummarization	hasDerivation
combinationFrom	isDerivationOf
hasCopy T	hasDerivation, sameCapabilityAs
hasSelection	hasDerivation, hasPart
isStandInOf	isDerivationOf, overlappingCapability-With
isCopyOf T	isDerivationOf, sameCapabilityAs
hasInterpretation	hasDerivation
isSelectionOf	isDerivationOf, isPartOf
isInterpretationOf	isDerivationOf
processedFrom	isDerivationOf

refactoredFrom	processedFrom
hasCache	hasSnapshot, hasStandIn
hasExample	hasSelection
isInferenceOf	isInterpretationOf
cleanedInto	overlappingCapabilityWith, processedInto
hasReification	processedInto
hasComputation	processedInto
isAnonymizedOf	isStandInOf, processedFrom
cleanedFrom	overlappingCapabilityWith, processed-From
remodelledFrom	processedFrom, samePopulationAs
hasAnonymized	hasStandIn, processedInto
hasSnapshot	hasCopy, versionOf
isExtractionOf	isInterpretationOf
isSnapshotOf	isCopyOf, versionOf
refactoredInto	processedInto
isCacheOf	isSnapshotOf, isStandInOf
hasExtraction	hasInterpretation
hasInference	hasInterpretation
isComputationOf	processedFrom
remodelledTo	processedInto, samePopulationAs
isExampleOf	isSelectionOf
optimizedInto	overlappingCapabilityWith, processedInto
optimizedFrom	overlappingCapabilityWith, processed-From
isReificationOf	processedFrom
hasStatistic	describedBy, hasComputation
isStatisticOf	describes, isComputationOf

3.2.3 Adjacencies

The top relation `adjacentTo` represents proximity between two datanodes in a data container (catalogue or dataset).

Table 3.5: Properties in the adjacency branch

relation		**sub property of**
adjacentTo	S T	relatedWith
disjointPartWith	S	adjacentTo
attachedTo		adjacentTo
hasAttached		adjacentTo
isAnnotationOf		about, attachedTo
disjointPortionWith	S	differentPopulationFrom, disjointPartWith
disjointSectionWith	S	differentVocabularyFrom, disjoint-PartWith
hasAnnotation		hasAttached, metadata

Proximity may result from being parts of the same dataset - `disjointPartWith`. Another case of proximity is between an object and its attachment - `hasAttached` / `attachedTo`. The `hasAnnotation` / `isAnnotationOf` relation, mentioned above, specialises also `hasAttached` / `attachedTo`.

Table 3.5 shows the relations in this branch.

3.2.4 Capabilities (overlapping and different)

Capability is intended as *"the power or ability to generate some outcome"*[4], and it is covered with two separate branches starting from `overlappingCapabilityWith` and `differentCapabilityFrom` respectively. Two data nodes may have similar potential. This may refer to any kind of feature, be it structural (e.g., they share schema elements), physical (e.g., they are both in XML) or related to the domain (they both talk about Music Artists) - to name but a few examples. This relation is intentionally left abstract since there might be many different ways to express capabilities. Extensions for specific use cases can of course be made as specialisations of this relation. The Datanode Ontology therefore only contains a few general cases: `overlappingVocabularyWith` and `overlappingPopulationWith`, both leading to `redundantWith`, `sameCapabilityAs` and `duplicate` - all describing a similar phenomenon with different intentions. Under this scope we also positioned `optimizedFrom` / `optimizedInto` - to state the empowerment of an existing capability; and `cleanedFrom` / `cleanedInto` - to represent the result of a process aimed to make emerge a potential capability

[4]Definition from `http://en.wiktionary.org/wiki/capability`.

whilst maintaining another fundamental one.

On the contrary, two datanodes may have different potential. For example, two data nodes use different vocabularies (`differentVocabularyFrom`, or `disjointSectionWith`, when parts of the same datanode) or have different population (`differentPopulationFrom`, or `disjointPortionWith`). When both vocabularies and populations are different it is possible that two datanodes have disjoint capabilities - `disjointCapabilityWith`.

Having an overlapping population can be the implication of a part-whole relation: `hasPart` / `isPartOf`. A distinction is done via `hasSection` / `isSectionOf` and `hasPortion` / `isPortionOf`. We call *section* the partition of a datanode by the means of a set of attributes (with no information about its population), while *portion* is a partition done by keeping a part of the population (with no information about the attributes). In the extreme case, hasSection can be further specialised into `hasIdentifiers` / `identifiersOf`. A portion can also be a sample - `hasSample` / `isSampleOf`. Deriving by selection is also a way of isolating a part of the source, thus `hasSelection` / `isSelectionOf` appears also on this branch, with the sub-property `hasExample` / `isExampleOf`.

Overlapping capability may be implied by being a *version*. This property implies a temporal relation between two data nodes that are meant to be the same at a different point in time. We find under this category relations such as `newerVersionOf` / `olderVersionOf` and `nextVersionOf` / `previousVersionOf`.

`versionOf` itself is symmetric and does not specify a direction. It is not transitive. While it can be argued that the identity of something tracked over time should not change, thus implying transitivity, we want to support the case when a datanode has more than one single following version (branching).

Table 3.6 shows the implications in the overlapping capabilities branch, while Table 3.7 the ones in the difference capabilities branch.

Table 3.6: Properties in the overlapping capabilities branch

relation	**sub property of**
overlappingCapabilityWith S	relatedWith
cleanedInto	overlappingCapabilityWith, processedInto
cleanedFrom	overlappingCapabilityWith, processed-From
hasStandIn	hasDerivation, overlappingCapabilityWith

overlappingPopulationWith		overlappingCapabilityWith, sharesInterpretationWith
overlappingVocabularyWith		overlappingCapabilityWith, sharesInterpretationWith
isStandInOf		isDerivationOf, overlappingCapabilityWith
versionOf	S	overlappingCapabilityWith
optimizedInto		overlappingCapabilityWith, processedInto
optimizedFrom		overlappingCapabilityWith, processedFrom
hasCache		hasSnapshot, hasStandIn
isAnonymizedOf		isStandInOf, processedFrom
hasAnonymized		hasStandIn, processedInto
isPartOf	T	overlappingPopulationWith
hasSnapshot		hasCopy, versionOf
samePopulationAs	T	overlappingPopulationWith
links		overlappingPopulationWith, references
isSnapshotOf		isCopyOf, versionOf
redundantWith	T	overlappingPopulationWith, overlappingVocabularyWith
linkedBy		overlappingPopulationWith, referencedBy
sameVocabularyAs	T	overlappingVocabularyWith
newerVersionOf	T	versionOf
isCacheOf		isSnapshotOf, isStandInOf
olderVersionOf	T	versionOf
hasPart	T	overlappingPopulationWith
hasExample		hasSelection
remodelledFrom		processedFrom, samePopulationAs
hasUpdatedVersion		hasUpdate, previousVersionOf
identifiersOf		isSectionOf
nextVersionOf	F	newerVersionOf
previousVersionOf	I	olderVersionOf
hasIdentifiers		hasSection

isSampleOf		isPortionOf
hasPortion	T	hasPart
hasSection	T	hasPart
isUpdatedVersionOf		isUpdateOf, nextVersionOf
hasCopy	T	hasDerivation, sameCapabilityAs
hasSelection		hasDerivation, hasPart
hasSample		hasPortion
sameCapabilityAs	T	samePopulationAs, sameVocabularyAs
isPortionOf	T	isPartOf
remodelledTo		processedInto, samePopulationAs
isCopyOf	T	isDerivationOf, sameCapabilityAs
isSectionOf	T	isPartOf
isExampleOf		isSelectionOf
isSelectionOf		isDerivationOf, isPartOf
duplicate	S T X	sameCapabilityAs

Table 3.7: Properties in the different capabilities branch

relation		**sub property of**
differentCapabilityFrom	S	relatedWith
differentPopulationFrom	S	differentCapabilityFrom
differentVocabularyFrom	S	differentCapabilityFrom
disjointPortionWith	S	differentPopulationFrom, disjointPartWith
disjointSectionWith	S	differentVocabularyFrom, disjoint-PartWith
disjointCapabilityWith	S	differentPopulationFrom, differentVocab-ularyFrom

3.2.5 Shared interpretation

This branch is dedicated to the possibility that datasets might share the same interpretation. This is designed to capture the possibility that a datanode might contribute to inferences that can be made in another one. Two datanodes *might* be "understood" together, i.e. its knowledge can be compared, or the interpretation (inferences) of one may affect the interpretation (inferences) of

another. The top relation `sharesInterpretationWith` is transitive and symmetric. Table 3.8 shows the implications in the shared interpretation branch. The explanations of the reasons why the immediate sub-relations imply a shared interpretation are summarized in Table 3.9, and discussed in the following part of this section.

Table 3.8: Properties in the shared interpretation branch

relation		**sub property of**
sharesInterpretationWith	S T	relatedWith
isVocabularyOf		sharesInterpretationWith
inconsistentWith	S	sharesInterpretationWith
overlappingPopulationWith		overlappingCapabilityWith, sharesInterpretationWith
consistentWith	S	sharesInterpretationWith
hasVocabulary		sharesInterpretationWith
overlappingVocabularyWith		overlappingCapabilityWith, sharesInterpretationWith
references		sharesInterpretationWith
referencedBy		sharesInterpretationWith
hasUpdate		sharesInterpretationWith
isUpdateOf		sharesInterpretationWith
schemaUsedBy		referencedBy
descriptorsOf		isVocabularyOf
hasDatatypes		hasVocabulary
hasUpdatedVersion		hasUpdate, previousVersionOf
isPartOf	T	overlappingPopulationWith
samePopulationAs	T	overlappingPopulationWith
links		overlappingPopulationWith, references
hasTypes		hasVocabulary
isChangeOf	F	isUpdateOf
isUpdatedVersionOf		isUpdateOf, nextVersionOf
redundantWith	T	overlappingPopulationWith, overlappingVocabularyWith
typesOf		isVocabularyOf
datatypesOf		isVocabularyOf

usesSchema		references
linkedBy		overlappingPopulationWith, referencedBy
sameVocabularyAs	T	overlappingVocabularyWith
hasChange	I	hasUpdate
isDependencyOf		referencedBy
hasDescriptors		hasVocabulary
hasPart	T	overlappingPopulationWith
hasDependency		references
hasCache		hasSnapshot, hasStandIn
hasExample		hasSelection
isDeletionOf		isChangeOf
remodelledFrom		processedFrom, samePopulationAs
relationsOf		descriptorsOf
hasDeletion		hasChange
identifiersOf		isSectionOf
hasSnapshot		hasCopy, versionOf
hasRelations		hasDescriptors
hasIdentifiers		hasSection
isSampleOf		isPortionOf
attributesOf		descriptorsOf
hasPortion	T	hasPart
isSnapshotOf		isCopyOf, versionOf
hasSection	T	hasPart
hasAttributes		hasDescriptors
hasCopy	T	hasDerivation, sameCapabilityAs
hasSelection		hasDerivation, hasPart
hasAddition		hasChange
hasSample		hasPortion
isCacheOf		isSnapshotOf, isStandInOf
sameCapabilityAs	T	samePopulationAs, sameVocabularyAs
isAdditionOf		isChangeOf
isPortionOf	T	isPartOf
remodelledTo		processedInto, samePopulationAs

isCopyOf	T	isDerivationOf, sameCapabilityAs
isSectionOf	T	isPartOf
isExampleOf		isSelectionOf
isSelectionOf		isDerivationOf, isPartOf
duplicate	S T X	sameCapabilityAs

Being (in)consistent. A shared interpretation obviously derives when we describe two datanodes to be *consistent* or *inconsistent* with each other. This aspect is covered by two properties: `consistentWith` and `inconsistentWith`. We intend (in)consistency in a broad sense: two datanodes are (in)consistent because they are (not) compatible in some fundamental respect. For example, `(in)consistentWith` may be used to indicate a data node that should (not) be used together. In some settings, it may be useful to specify that some datanodes are consistent with others, eventually to state that a datanode is `consistentWith` itself. It is also possible that a datanode is `inconsistentWith` itself, meaning that it is wrong, buggy or somehow corrupted.

Being an update. A data node may be related to another because it contributes to improve its validity or currency. `hasUpdate` (and its inverse `isUpdateOf`) has this role. This property can be specialised to indicate a datanode that is meant to substitute the original: `hasUpdatedVersion` / `isUpdatedVersionOf` (that are also under the `hasVersion` umbrella). While the updating (as activity) can be seen as a way of versioning, the two concepts are different when modelled as relations between datanodes. An update may not replace its target datanode. It is the case when the update is meant to only modify the data node: `hasChange` / `isChangeOf` and its sub-properties `hasAddition` / `isAdditionOf` and `hasDeletion` / `isDeletionOf`.

Being referenced. A datanode includes a mention to something which is another datanode. This branch covers linking (as intended in the context of linked data) - `links` / `linkedBy`; as well as dependencies between data objects (`hasDependency` / `isDependencyOf`) and between data and a schema definition, for example an ontology (`usesSchema` / `schemaUsedBy`).

Being a vocabulary. The range of hasVocabulary is a datanode which enumerates a set of terms that are all used by the subject data node as identifiers (inverse is `isVocabularyOf`), to name structural elements like `hasDescriptors` / `descriptorsOf`. These can be attributes, relations, or to link a datanode to its types - `hasTypes` / `typesOf` or datatypes with `hasDatatypes` / `datatypesOf`. This is different from the case of a relation with a schema (under "references"): a schema is defined independently from the actual data.

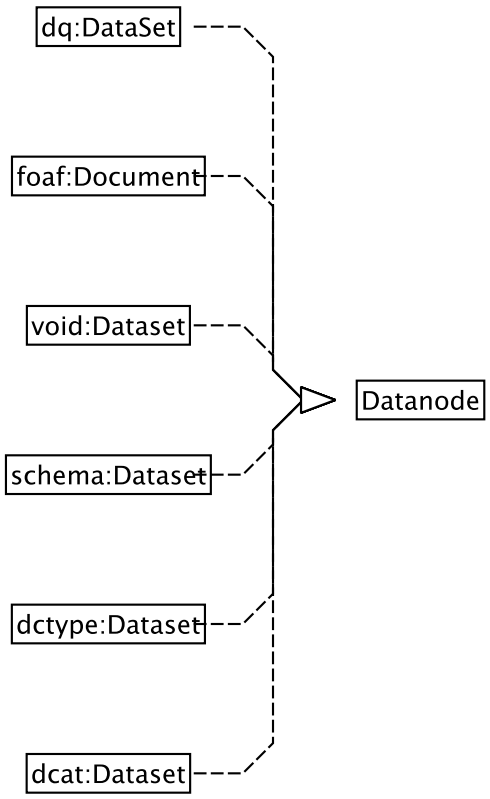


Figure 3.3: Datanode: alignments to classes of Semantic Web ontologies. All these classes are sub classes of Datanode.

3.2.6 Alignments with Semantic Web ontologies

As also discussed in Section 3.5, there exist a number of Semantic Web ontologies that represent aspects of data artefacts tailored to common use cases. Datanode is indeed related to DCAT [Erickson and Maali (2014)], Prov-O [Lebo et al. (2013)], VoID [Alexander and Hausenblas (2009)] and VOAF [Vandenbussche and Vatant (2011)] by aligning a number of Classes to the class `Datanode`, and then by placing the relations that occur between datanodes in the property hierarchy that extend the top layer of the ontology. Figure 3.3 displays a number of classes that we specify to be specialisations of Datanode, while Table 3.10 lists alignments between Datanode and the related relations.

The main focus of DCAT is to provide the basic tools to classify datasets in repositories using concepts, pointing to publishing web pages and distinguishing a dataset and its distributions (concrete instances). The property `dcat:record` links a catalogue to a catalogue record that

Table 3.9: Relations that imply a shared interpretation.

:isUpdateOf	being an update implies that two nodes can be compared to understand how a specific information evolved/changed
:hasUpdate	similarly, know that some data has an update implies that its validity or timeliness is compromised
:overlapping-PopulationWith	If two data nodes contains the same entities, certainly they might be read together
:overlapping-VocabularyWith	If two data nodes express the information with the same properties or types then they may be combined
:inconsistentWith and :consistentWith	Being (in)consistent is defined as being (in)compatible, thus imply sharing something at a fundamental level.
:references and :referencedBy	If a node references another one, then its interpretation could be affected by the other (and vice versa).
:hasVocabulary and :isVocabularyOf	The understanding of the terms used as properties and types surely affect the interpretation of the data.

has `foaf:primaryTopic` a given `dcat:Dataset`. The property `dcat:dataset` makes a direct link from the catalogue to each dataset that is described. The Prov-O ontology includes the concepts of derivation and revision, and both are positioned in the hierarchy of Datanode. Section 3.4 will show how this connection can have interesting consequences in the inferred model. The VOAF vocabulary enumerate a number of ways datanodes that are schemas can be linked together[5], most of them imply derivation and reference to be in place.

3.2.7 Compliance with OWL2 profiles

Table 3.11 shows the compatibility of the whole Datanode Ontology with relation to OWL2 profiles. The Datanode Ontology is in the OWL 2 DL profile.

It is not in the OWL-RL profile because it uses `owl:IrreflexiveProperty` to express the relation `duplicate`. This is the only axiom that prevents the ontology to fall into the OWL-RL profile. In cases when it is desirable, this axiom can be easily removed. The ontology is not in the OWL-QL profile because of its use of properties of type `owl:TransitiveProperty`

[5]It is worth to remember that the meaning we give to Vocabulary in Datanode is being an enumeration of symbols all used in some data. We prefer to call the objects of the VOAF ontology "schemas", or ontologies.

Table 3.10: Properties of some Semantic Web ontologies aligned to Datanode relations using `rdfs:subPropertyOf`.

relation	**sub property of**
http://purl.org/dc/terms/references	references
http://purl.org/dc/terms/subject	about
http://purl.org/vocommons/voaf#extends	references
http://purl.org/vocommons/voaf#extends	isDerivationOf
http://purl.org/vocommons/voaf#generalizes	references
http://purl.org/vocommons/voaf#generalizes	isDerivationOf
http://purl.org/vocommons/voaf#metadataVoc	usesSchema
http://purl.org/vocommons/voaf#reliesOn	references
http://purl.org/vocommons/voaf#similar	overlappingCapabilityWith
http://purl.org/vocommons/voaf#specializes	references
http://purl.org/vocommons/voaf#specializes	isDerivationOf
http://purl.org/vocommons/voaf#usedBy	references
http://rdfs.org/ns/void#subset	hasPart
http://rdfs.org/ns/void#target	links
http://rdfs.org/ns/void#vocabulary	usesSchema
http://www.w3.org/ns/dcat#dataset	describes
http://www.w3.org/ns/dcat#record	hasPortion
http://www.w3.org/ns/prov#wasDerivedFrom	isDerivationOf
http://www.w3.org/ns/prov#wasRevisionOf	isUpdatedVersionOf
http://xmlns.com/foaf/0.1/primaryTopic	describes

Table 3.11: Compliance with OWL2 profiles.

Profile	**?**
OWL 2	✓
OWL 2 DL	✓
OWL 2 RL	x
OWL 2 EL	x
OWL 2 QL	x

(for example, `hasPart` is transitive) and of type `owl:InverseFunctional` (for example, `previousVersionOf`). For similar reasons, it is not in the OWL-EL profile: It includes `owl:inverseOf` (many relations have their inverse declared) and properties of type `owl:SymmetricProperty` (eg, `differentCapabilityFrom`) as well as `owl:FunctionalProperty` and `owl:InverseFunctionalProperty`[6].

3.3 Describing systems: evaluation in practice

In this section we describe how the Datanode ontology can be used to describe eight different systems. Six of these applications are the ones used in the survey part of the methodology. In the testing phase we added two more applications, Rexplore and IBM Watson. We present this as an evaluation of both the abstract conceptualisation - foundation of the ontology, and of the applicability of the ontology itself.

The preparation of each test case followed a fixed procedure:

1. isolate portions of text describing the system (similarly to what happened in the initial phase of our methodology, see Section 3.1);
2. identify data source(s) and data output and model them as datanodes;
3. connect the two using the properties defined in the ontology to express the data flow of the system, by adding any necessary datanode in between.

The use cases are grouped following the definition of Semantic Web Applications mentioned in Section 3.1.

SWAs make use of information from sources they do not control. **EventMedia** [Khrouf and Troncy (2012)] is a a web-based system that exploits real-time connections to enrich content describing events and associate it with media objects[7]. The emphasis here is on the linkage with multiple sources for collecting event data and multimedia objects. Figure 3.4 displays how data in EventMedia is taken from three public event directories and is enriched with data from sources such as BBC or Foursquare [Khrouf and Troncy (2012)]. The information is then presented using the LODE ontology.

Similarly, **Yokohama Art Spot** relies on linked data to present the cultural offers of artistic institutions spread in the area of Yokohama [Matsumura et al. (2012)] - see Figure 3.5 .

[6]The compatibility with OWL 2 profiles has been tested practically with the service `http://mowl-power.cs.man.ac.uk:8080/validator/`.

[7]See `http://eventmedia.eurecom.fr/`.

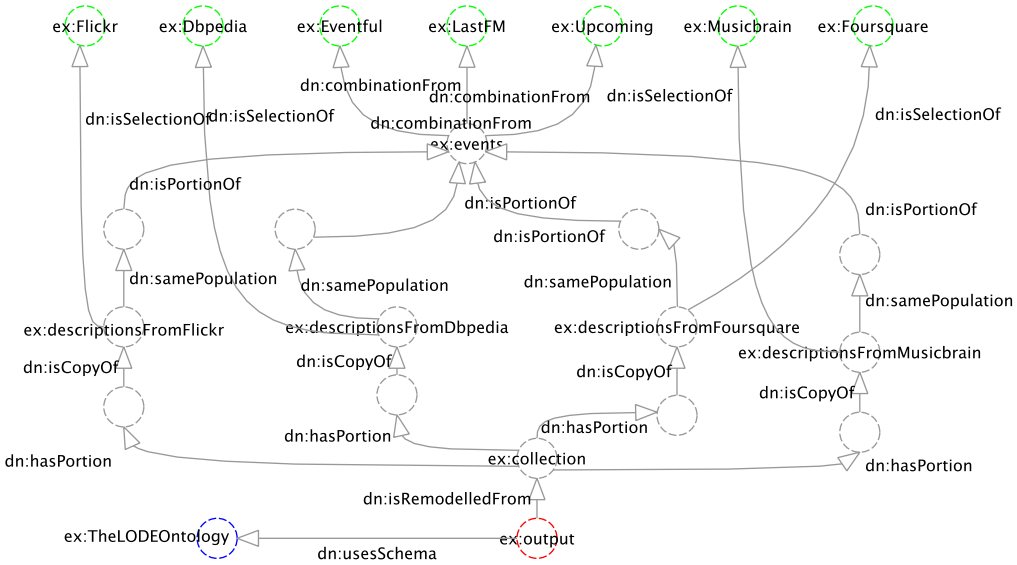


Figure 3.4: Description of the reuse of sources in the EventMedia mash-up [Khrouf and Troncy (2012)]. The input sources are the top nodes. The node at the bottom depicts the output data, which is a remodelling of the data collected from various sources according to a specific schema. Used branches: reference, derivation, partition, capability.

SWAs manipulate information to produce new knowledge. **DBRec** is a music recommendation system built using DBPedia [Passant (2010)]. The paper, among other aspects, describes the preparation of the data done to maximise the efficiency of the Linked Data Semantic Distance algorithm. The Datanode graph depicted in Figure 3.6 shows how a process that goes from the DBPedia dataset to an optimised and curated dataset can be formalised. **SPUD** exploits semantic technologies *"to obtain business results in an environment with hundreds of heterogenous real datasets coming from different data sources and spanning multiple domains"* [Kotoulas et al. (2014)]. The system is a web application with several functionalities. One of them is the *"incremental semantic lifting of data. [...] We capture provenance (using the latest W3C working draft) by making views over datasets immutable"* [Kotoulas et al. (2014)]. It is quite easy to show this engineering solution purely at the knowledge level using datanodes (see Figure 3.7). In the same application, some data are published in a slightly changed fashion in order to protect sensible information. Figure 3.8 represents this solution. The published data can be described as composed by two sections, one containing all the non private information from the database, the second being an anonymisation of the private data (the new knowledge, in our definition). **DiscOU** is presented as a *"discovery engine for Open Educational Resources Using Semantic Indexing and Relationship Summaries"* [d'Aquin et al. (2012)]. The indexing component stores DBPedia entities extracted from the text of an educational resource. Giving a resource as input the engine performs a similarity search returning

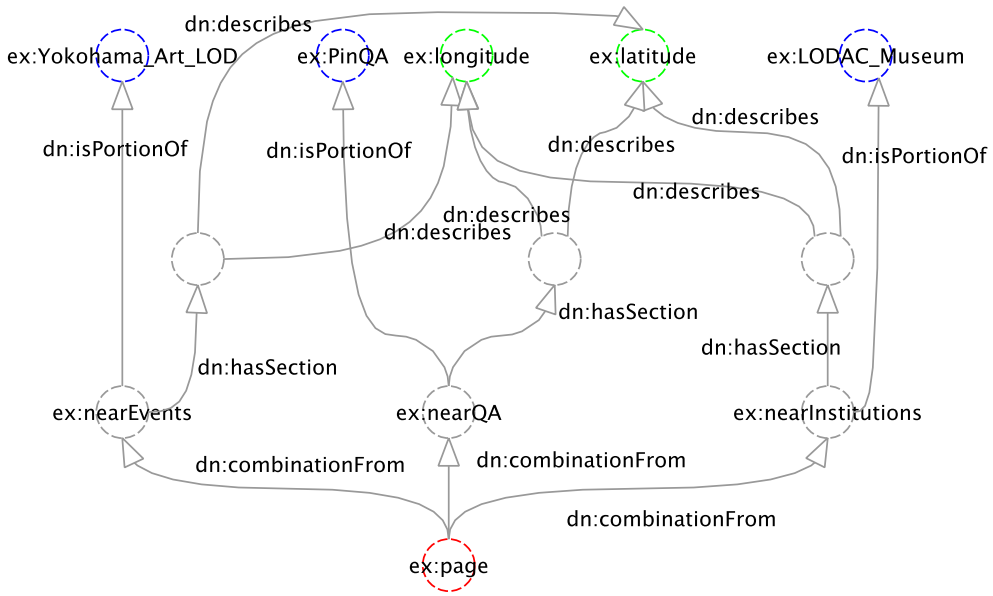


Figure 3.5: Yokohama Art Spot [Matsumura et al. (2012)]. Starting from the user's location, the system aggregates information from artistic events and museums nearby, packing it in a single web page. Used branches: derivation, partition.

.

the best similar resources. This process is described in Figure 3.9.

SWAs process the resulted data for presentation in a new form. This last part of the evaluation is dedicated to show how the visible results of a system's behaviour can be sustained by a formal description with Datanode. **Aemoo** is presented as a web application for exploratory search over the Semantic Web [Musetti et al. (2012)]. The input requested is the name of a Semantic Web entity (as string) and the output is a summarisation of the relevant knowledge about the entity extracted from DBPedia and other linked data sources and arranged according a set of Knowledge Patterns [Musetti et al. (2012)] mapped to entity types. We can see in Figure 3.10 a description of the system according to Datanode. The output node `dn:resourceSummary` is `dn:isSummarizationOf` the input, `dn:remodelledFrom` a node which in turn `dn:isPortionOf` DBPedia. The output `dn:usesSchema` the knowledge pattern `dn:isDescriptionOf` the resource's primary type, a dn:isSelectionOf the types of the resource, from the set of DBpedia types. More precisely, the knowledge pattern is one of a set of summarisation methods which are a `dn:combinationFrom` knowledge patterns and DBPedia types. It is clear that we have abstracted from the implementation of Aemoo, while we have now a better understanding of its output. Finally, the output is presented as `dn:isStandInOf` the data about the input resource.

Rexplore *"integrates statistical analysis, semantic technologies, and visual analytics to provide*

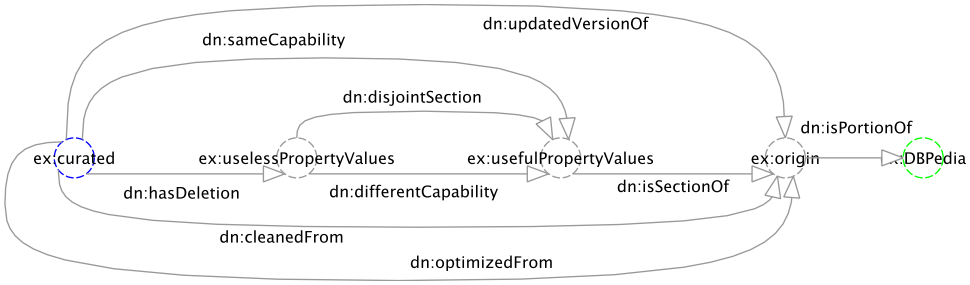


Figure 3.6: Synthetic description of the DBRec data preparation process. Used branches: derivation, partition, update, consistency and capability.

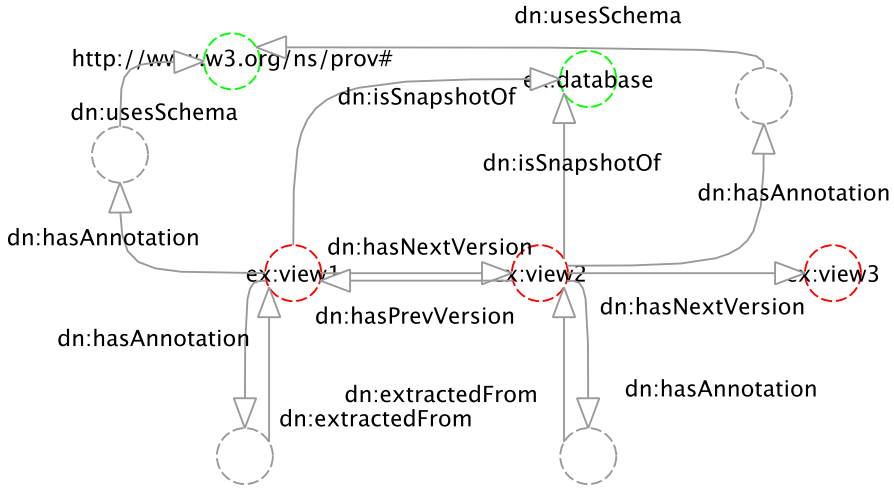


Figure 3.7: In Spud, the database is versioned. Each view is a snapshot of the data, annotated with provenance information. Semantic lifting is executed and stored in the views as data enrichment (they are the nodes at the bottom, using the pattern hasAnnotation-isExtractedFrom). Uses: reference, adjacency, version, derivation

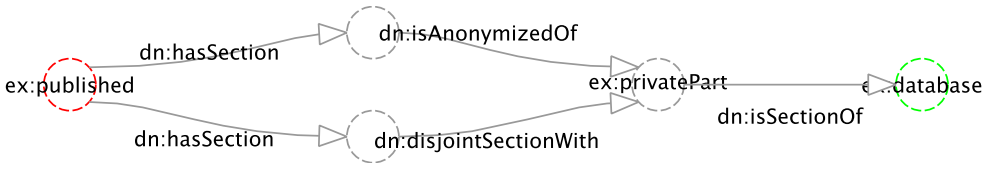


Figure 3.8: In Spud, data is processed to hide sensible information. Information publicly shared contains the part of the database with no privacy concerns as well as some data which is preprocessed. Uses: partition, capability, derivation.

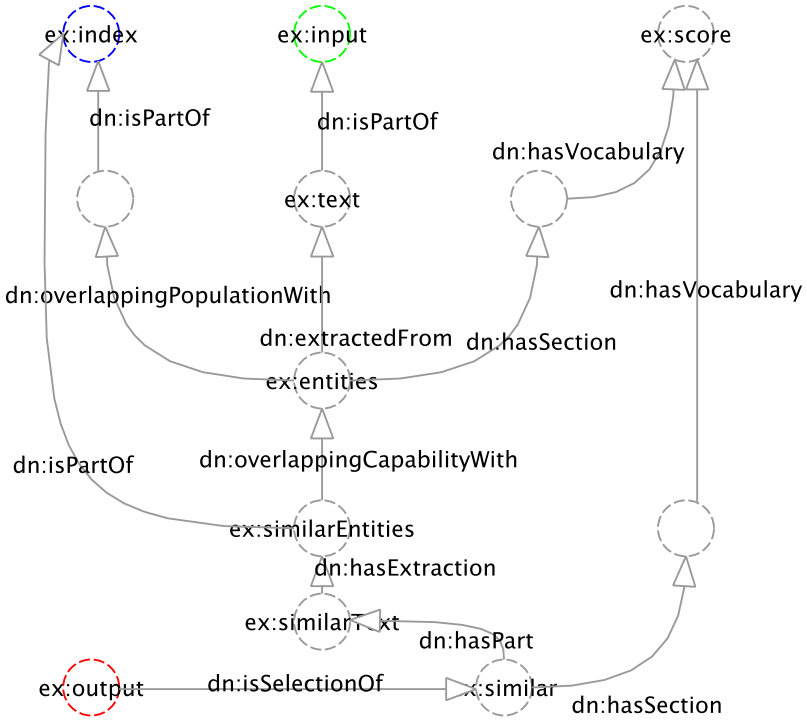


Figure 3.9: The similarity search process in DiscOU. This example shows the usage of derivation, partition and capability.

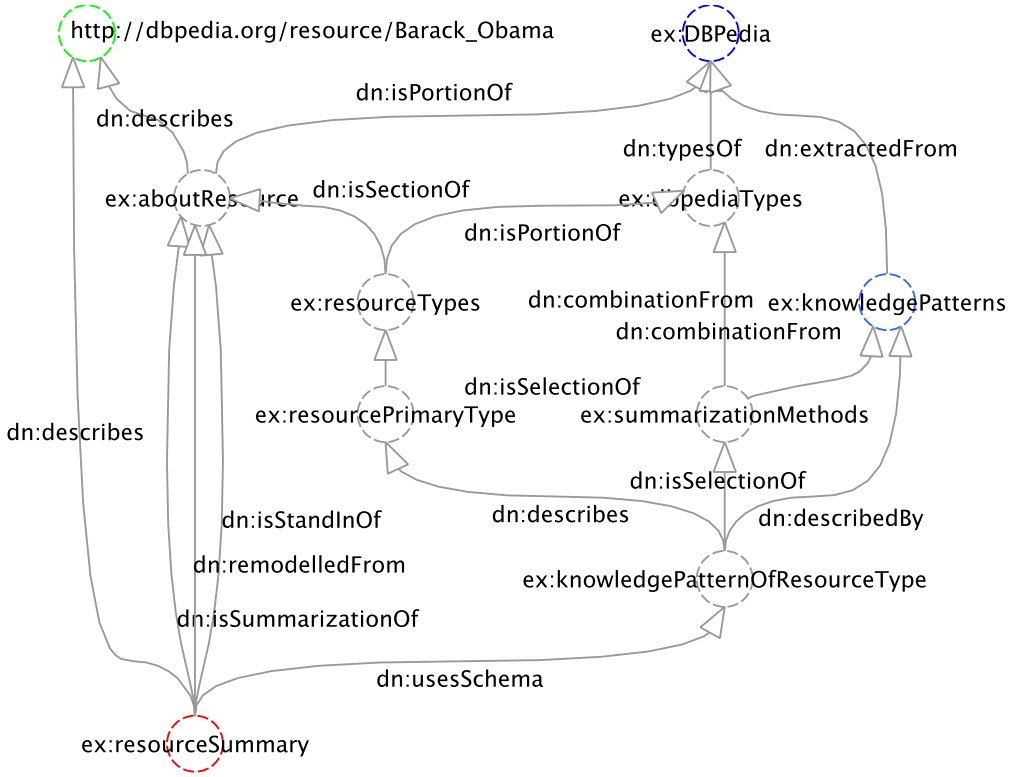


Figure 3.10: Description of AEMOO. The input is a resource described in DBPedia. The system outputs a resource summary. Used branches: meta, reference, derivation, partition, vocabulary.

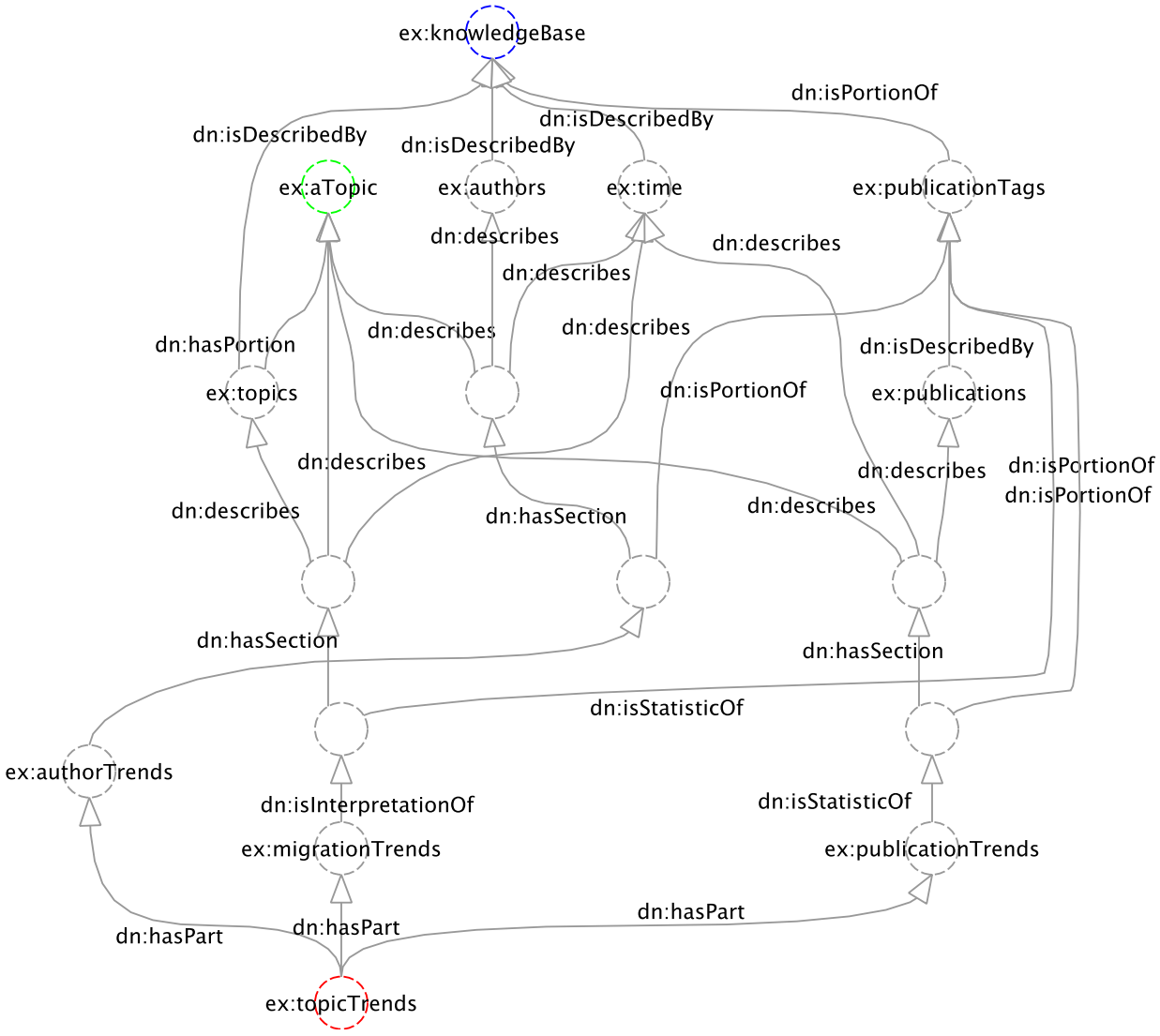


Figure 3.11: The visualised information about the trends of `:aTopic` in Rexplore. The web page includes publication trends, author trends and migration trends. Uses: meta, derivation, partition.

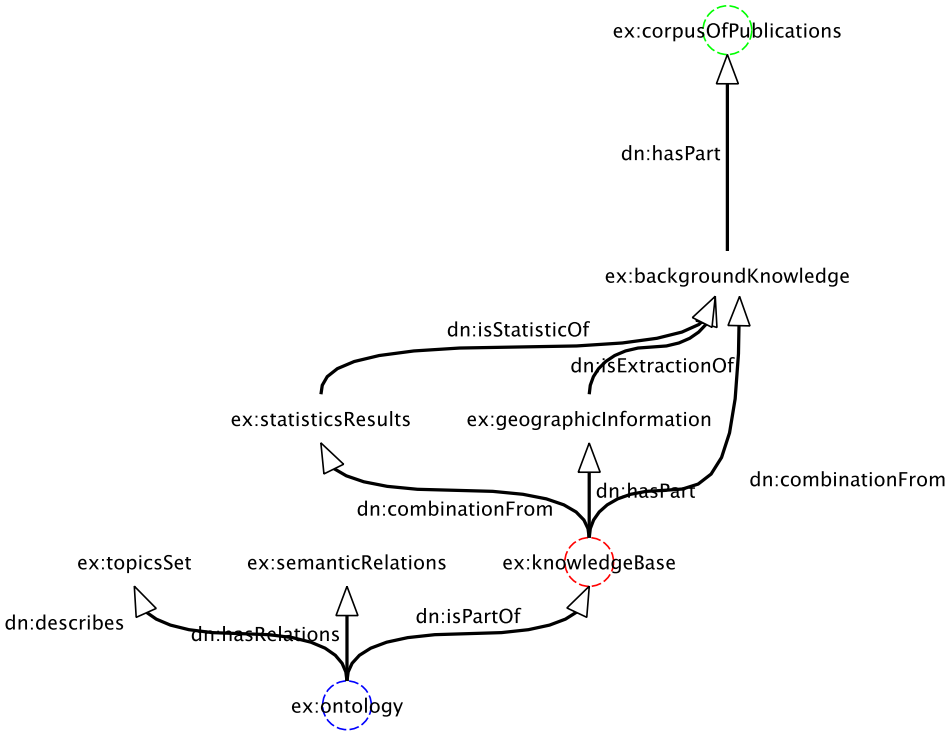


Figure 3.12: The knowledge base of Rexplore. The input is a corpus of publications, in green. Output is red.

effective support for exploring and making sense of scholarly data" [Osborne et al. (2013)]. Figure 3.11 shows a diagram of the visualised information about Topic's trends. Publication trends is the number of publications associated with a semantically enriched topic on a timeline; similarly, author trends are the number of associated authors. Migration trends are the number of estimated migrations between two topics computed by analysing the shifting in authors' interest. Figure 3.12 shows instead the knowledge components of Rexplore.

IBM Watson is the well known computer that became famous after winning the *Jeopardy!* contest. Figure 3.13 shows the data flow of the system as it is described in the Wikipedia page [Wikipedia (2014)].

We have been able to express with Datanode all the data flows of the Semantic Web Applications collected. Table 3.12 shows the coverage of the Datanode ontology branches on each application considering not only the descriptions reported here but all the use cases evaluated.

3.4 Applying datanode: an exemplary scenario

In this section we describe a protypical use case, focused on understanding change propagations in a data hub. A datahub contains a number of datasets owned and managed by different actors. Nevertheless, many datasets "depend" on others in their interpretation, thus need to be reviewed (for example updated) if some change in the features of their reference dataset occurs.

The UK Higher Education Statistics Agency (HESA) publishes the Unistats dataset[8]. Each September HESA publishes a new version of the dataset, with changes in the data schema, containing improvements and new data. The format of the data is an XML serialization based on the Key Information Set (KIS) [Davies (2012); Renfrew et al. (2010)].

The LinkedUp Project was a FP7 Support Action "which pushes forward the exploitation and adoption of public, open data available on the Web, in particular by educational organisations and institutions"[9]. The project published a catalogue of datasets as Linked Data [d'Aquin et al. (2013)]. For example, Open Data from the Italian National Research Council (CNR) [Gangemi et al. (2011)] and from the PROD project[10] are registered in the same catalogue, as well as many other datasets. Since 2012, the catalogue includes a Linked Data version of the Unistats database (Unistats LD). Unistats LD is a *remodelling* of the Unistats XML published by HESA, using a schema (KIS-LD) that is mostly based on the RDF Data Cube Vocabulary [Reynolds and Cyganiak (2014b)] and is itself a remodelling of the original schema (KIS). The PROD dataset, however,

[8] See `https://www.hesa.ac.uk/unistatsdata`.

[9] See `http://linkedup-project.eu/about/`.

[10] PROD is a directory and monitoring tool for JISC funded projects. See `http://prod.cetis.ac.uk/`.

Table 3.12: Datanode branches, coverage on the evaluated use cases.

	EventMedia	Yokohama Art Spot	DBRec	Spud	DiscOU	Aemoo	Rexplore	IBM Watson
meta						x	x	
reference	x			x		x		
derivation	x	x	x	x	x	x	x	x
adjacency				x	x			
partition	x	x	x	x	x	x	x	x
vocabulary	x				x	x		
version				x				
update			x					
consistency			x					
capability	x		x	x	x			x

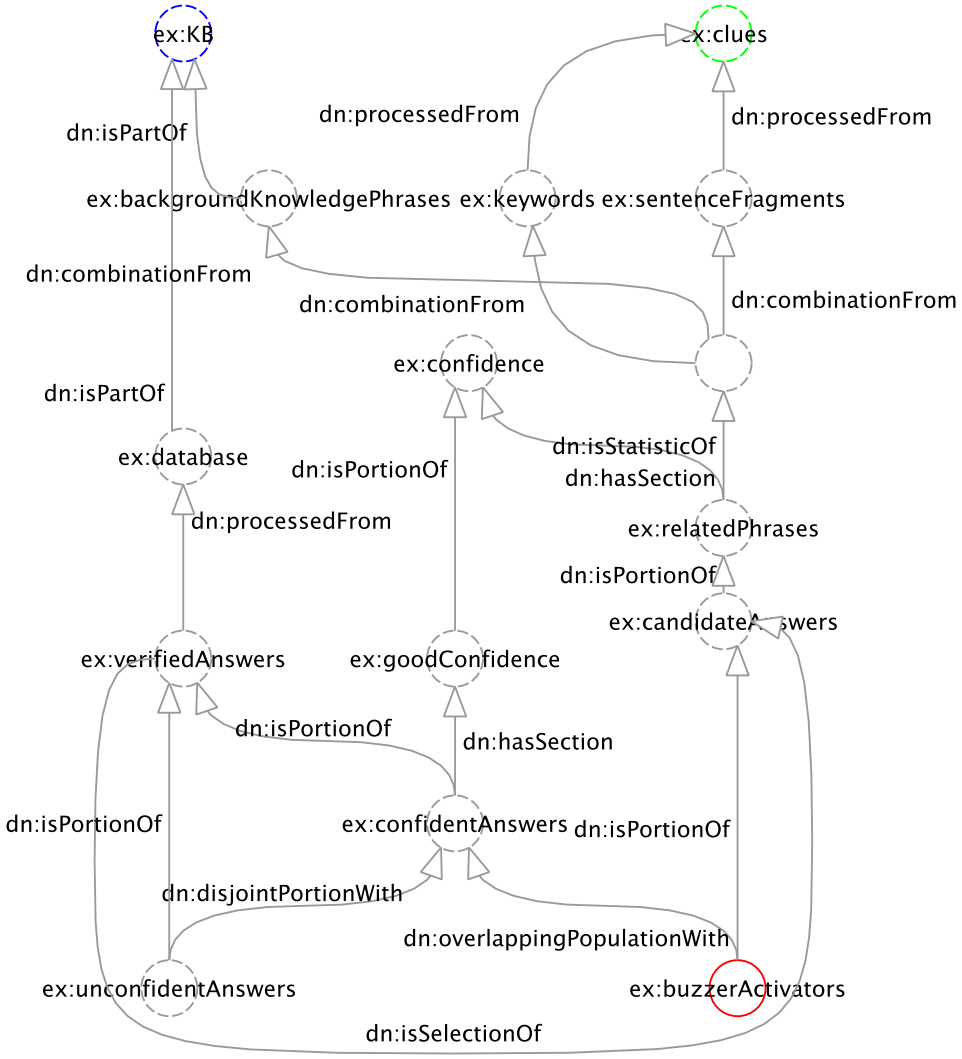


Figure 3.13: Watson at *Jeopardy!*, from the "clues" to the set of answers "buzzer activators". This has been designed from the description on Wikipedia [Wikipedia (2014)]. Uses: derivation, partition and capability.

contains `owl:sameAs` links to the institutions described in the Unistats LD dataset. The manager of the LinkedUp catalogue regularly pings the HESA web site for newer version of the data, and the software used to translate the information is kept up to date once a new version of the schema is published. The LinkedUp catalogue add a new graph every year, once the information for a new academic year is published, and keep the old data as historical database.

The Open University (OU)[11], like all UK universities, provide its KIS record to HESA, so that it can be combined with the data from other universities to produce the statistics published as the Unistats dataset. However, the university also wants to use the slice of the Unistats data that refers to the OU - Unistats OU. For this reason the *portion* of the latest Unistats LD that is about the OU is selected and published in the Linked Open Data portal of the university[12] [Zablith et al. (2012)] aside other institutional datasets, including taught courses and qualifications, among others.

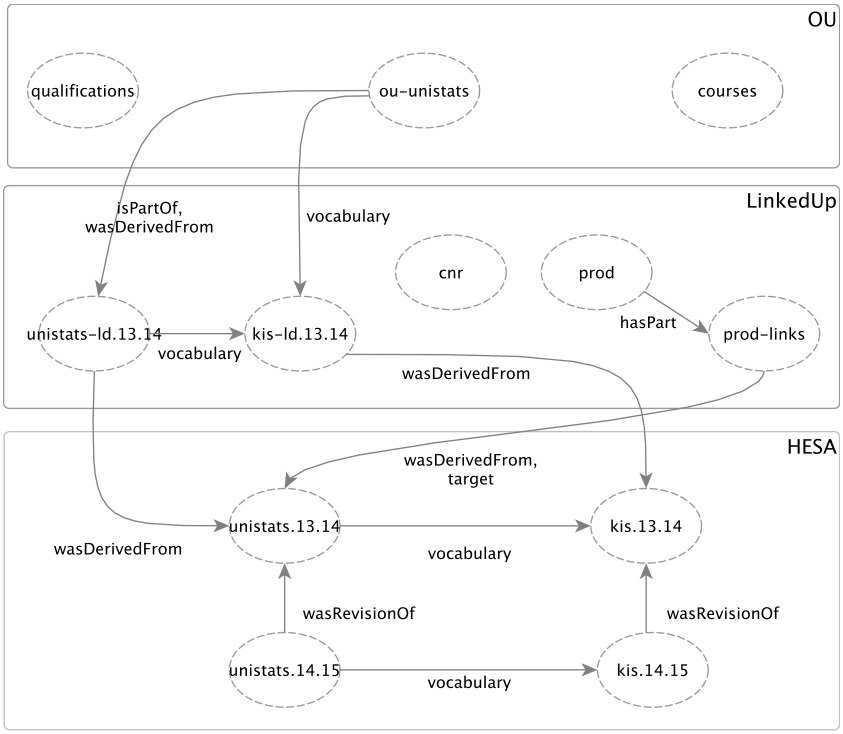


Figure 3.14: Summary of the scenario. Relation are expressed using Prov-O and VoID. In this picture we omit the 2012/2013 version of the datasets, for clarity.

To summarise:

– the `unistats` dataset uses `kis` as XML schema
– the `unistats-ld` uses `kis-ld`, both are the RDF remodelling of the first

[11] `http://www.open.ac.uk`
[12] `http://data.open.ac.uk`

- `prod` links to `unistats-ld`
- `ou-unistats` is a part of `unistats-ld`
- subsequent versions of `unistats` and `unistats-ld` are archived
- there are other datasets in `linkedup-cat` and `ou-cat`, like `cnr`, `ou-courses` and `ou-qualifications`

The scenario is depicted in Figure 3.14. Thank to Semantic Web ontologies, we can express the information about the catalogues (see Listing 3.1), as well as metadata about the data flow.

Listing 3.1: A snippet from the LinkedUp and OU catalogues description.

```
:linkedup-cat
 dcat:dataset :prod ;
 dcat:dataset :cnr ;
 dcat:dataset :unistats-ld.12.13 ;
 dcat:dataset :unistats-ld.13.14 ;
 ...

:ou-cat
 dcat:dataset :ou-courses ;
 dcat:dataset :ou-qualifications ;
 dcat:dataset :ou-unistats ;
 ...
```

Listing 3.2 shows the scenario described with the W3C Prov-O ontology and VoID[13]. For clarity, we omit part of the provenance information, for example the type declarations and the statements that relate the datasets to their respective owner using the `prov:wasAttributedTo` property.

Listing 3.2: The scenario described with Prov-O and VoID.

```
:unistats.13.14
    prov:wasRevisionOf :unistats.12.13 .

:unistats-ld.12.13
    prov:wasDerivedFrom :unistats.12.13 .

:unistats-ld.13.14
    prov:wasDerivedFrom :unistats.13.14 ;
    prov:wasRevisionOf :unistats-ld.12.13 .

:prod dcterms:hasPart :prod-links .
```

[13]It can be noted that we are describing the Unistats XML datasets as being part of the Linked Data. They are not, being in a non-RDF format. However, this does not change the meaning of the scenario for our demonstration.

```
:prod-links
    dcterms:isPartOf :prod ;
    prov:wasDerivedFrom :unistats-ld.13.14 ;
    void:target :unistats-ld.13.14 .

:ou-unistats
    dcterms:isPartOf :unistats-ld.13.14 ;
    prov:wasDerivedFrom :unistats-ld.13.14 .

:kis.13.14
    prov:wasRevisionOf :kis.12.13 .
:kis-ld.12.13
    prov:wasDerivedFrom :kis.12.13 .
:kis-ld.13.14
    prov:wasDerivedFrom :kis.13.14 .

:unistats.12.13 void:vocabulary :kis.12.13 .
:unistats.13.14 void:vocabulary :kis.13.14 .
:unistats-ld.12.13
    void:vocabulary :kis-ld.12.13 .
:unistats-ld.13.14
    void:vocabulary :kis-ld.13.14 .
```

The scenario described above contains a number of different actors (HESA, LinkedUp, OU), actions that have dependencies (data updates) and data items that are transformed into others (`unistats`/`unistats-ld`), selected and moved (`ou-unistats`), and linked (`prod` / `unistats-ld`). The whole process can be traced using the W3C PROV model so the administrators can have understanding of how data is produced, published and consumed[14].

However, when the statistics of the new academic year (e.g. 2014/2015) are published by HESA, there are a number of items in the catalogues that are affected in terms of timeliness, at least, and that require intervention by the related managers:

1. the LinkedUp catalogue administrator need to know which datasets in its catalogue needs to be updated (Unistats LD versions should be, while CNR should not);
2. the LinkedUp developer will have to review and update the KIS-LD schema to inherit the changes in the structure of the KIS XML schema;
3. and needs to adapt the extraction software to use the new schema and generate the new dataset;
4. old datasets are kept as historical reference, but pointers should be added to the latest version;
5. the OU needs to replace their dataset, it has to be a selection of the last Unistats LD 2014/15, and not of the 2013/14;

[14]While we could have detailed activities using PROV-O extensions, we simplified the implementation using only binary relations to show how, even with this simplification, querying the provenance graph is not trivial.

6. the PROD dataset needs to be updated. The portion containing the links need to be recomputed towards the new version because organisations or URIs might have changed.

Listing 3.3 lists the triples that reflect the change in the datahub, and that enrich the provenance database (an excerpt has been shown in Listing 3.2). The manager can query the provenance database to detect a number of possible items that are affected by the publishing of the new KIS dataset. Figure 3.4 shows a possible query to detect items that share some information with the new Unistats LD dataset.

Listing 3.3: The information added to the dataflow description once the new Unistats data about 2014/2015 is published by HESA.

```
:unistats.14.15
    prov:wasRevisionOf :unistats.13.14 .

:kis.14.15
    prov:wasRevisionOf :kis.13.14 .

:unistats.14.15 void:vocabulary :kis.14.15 .
```

Listing 3.4: SPARQL query to select affected items from the Provenance dataset.

```
SELECT distinct ?node WHERE {
  {
    :unistats.14.15  prov:wasRevisionOf ?node
  } UNION {
    ?node prov:wasDerivedFrom* ?something .
    :unistats.14.15  prov:wasRevisionOf* ?something
  } UNION {
    ?node dcterms:hasPart ?apa .
    ?apa prov:wasDerivedFrom ?ste .
    ?ste prov:wasDerivedFrom ?ast .
    :unistats.14.15 prov:wasRevisionOf ?ast
  } UNION {
    :unistats.14.15 void:vocabulary ?sc .
    ?sc prov:wasRevisionOf* ?node
  }
}
```

However, we can abstract the above points to a general Competency Question (CQ). Theoretically, any item that might *share an interpretation with* the new coming dataset is potentially affected by this event. The CQ would be: *Which are the datasets that might share the same interpretation model of the new coming dataset?* With Datanode we can add an abstraction layer on top of the

provenance information. At the same time, we can specify many interesting relations that are not directly captured by existing information, for example the fact that `ou-unistats` is actually a selection of `unistats.13.14`, or that `prod` and `cnr` are adjacent, being member of the same catalogue (see Listing 3.5).

Listing 3.5: Extending the Provenance information with Datanode we can qualify the relations between data items further.

```
:unistats.13.14
    dn:isUpdatedVersionOf :unistats.12.13 .

:prod dn:links :unistats-ld.13.14 .
:prod dn:hasPart :prod-links .
:prod-links dn:isSelectionOf :unistats-ld.13.14 .
:prod dn:adjacentTo :cnr .

:unistats-ld.12.13
    dn:remodelledFrom :unistats.12.13 .
:unistats-ld.13.14
    dn:remodelledFrom :unistats.13.14 .

:unistats-ld.13.14
    dn:isUpdatedVersionOf :unistats-ld.12.13 .

:ou-unistats dn:isCopyOf [
    dn:isSelectionOf :unistats-ld.13.14 ] .

:kis.13.14
    dn:nextVersionOf :kis.12.13 .
:kis-ld.12.13
    dn:remodelledFrom :kis.12.13 .
:kis-ld.13.14
    dn:remodelledFrom :kis.13.14 .

:unistats.12.13 dn:usesSchema :kis.12.13 .
:unistats.13.14 dn:usesSchema :kis.13.14 .
:unistats-ld.12.13 dn:usesSchema :kis-ld.12.13 .
:unistats-ld.13.14 dn:usesSchema :kis-ld.13.14 .

:unistats.14.15
    dn:isUpdatedVersionOf :unistats.13.14 .
:kis.14.15
    dn:nextVersionOf :kis.13.14 .
:unistats.14.15 dn:usesSchema :kis.14.15 .
```

By specifying the relations between datanodes we can infer which nodes "share interpretation with" the new `unistats.14.15` node, which is the kind of abstraction needed by this use case. This relation will occur with datanodes such as `:prod`, but not with the node `:cnr`. In

fact, while both datasets are part of the same `dcat:Catalogue` (being `dcat:dataset` a sub property of `dn:describes`) and they are `dn:adjacentTo`, they do not necessarily share the same interpretation. Existing methods do not directly support the process and modelling described in this scenario, they can only be adapted with ad-hoc complex queries like the one shown in Figure 3.4. Figure 3.15 shows the result: Datanode nicely enhances the provenance information with an inferencing shortcut.

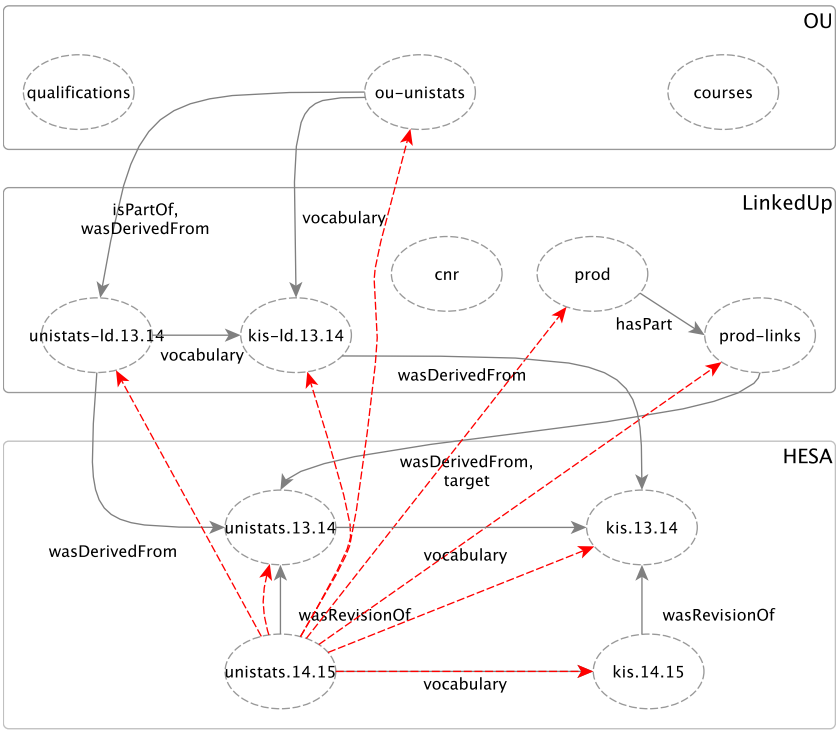


Figure 3.15: Adding Datanode to provenance information. The dotted red arcs show the Datanode relation `sharesInterpretationWith`.

Data artefacts are often part of composite scenarios. With Datanode, many interesting relations between datasets can be represented and inferred, supporting the management, monitoring and the understanding of (linked) data catalogues.

3.5 Related vocabularies and modelling practices

There are a number of Semantic Web ontologies that share the same domain as Datanode, and have contributed to the genesis of the ontology, among the ones discussed in the previous chapter, including FOAF, PROV, VoID, DCAT, and VOAF, just to mention some of them. All these Semantic Web Ontologies have been taken into account while designing Datanode. Their focus is to provide a

reusable schema for the description of certain classes of data objects, being data tables, vocabularies, datasets, catalogues or RDF documents. Datanode tries to design a general conceptualisation of the relations between data artefacts. In Section 3.2 we discussed how all of these ontologies align with Datanode. Here, we mention some approaches that relate particularly with our contribution. The DOOR ontology was implemented to represent the way Semantic Web ontologies can be related each other in ontology repositories [Allocca et al. (2009)]. Datanode is meant to address a very similar problem, but considered in a wider scenario.

The Asset Description Metadata Schema (ADMS) [Shukair et al. (2013)] is an RDF vocabulary proposed to be a common representation and exchange format of e-Government repositories. The Data Documentation Initiative (DDI) has developed a metadata specification for the social and behavioral sciences [Vardigan et al. (2008)]. The intent is to describe data sets resulting from social science research (like survey data). An RDF version of the vocabulary has been recently developed[15] to foster the publishing into the Web of Linked Data. ADMS and DDI provide reusable datasets as well as reusable metadata models and standards, while our contribution has at his core the analysis and organisation of the possible relations between data artefacts.

Research about workflow descriptions and provenance strongly relate with our work, especially if considering the methodology we used to build the ontology, starting from the analysis of data flows. For example, [Ram and Liu (2009)] define a set of use cases for provenance modelling from which they generalise a number of abstract requirements, including the value of making explicit the intent of the agents that make part to an activity, which semantically affects the result. The Datanode branch related to derivation of datasets also makes explicit the possible semantics of data manipulations. In [Hartig and Zhao (2010)] the authors develop a vocabulary to annotate data with provenance information, including metadata about the process, involved software and provider. The goal is to support the selection of linked data sources based on trustworthiness. Datanode can be used as a complement to provenance, to represent and reason on what are the implications of these aspects in terms of the relationships between the data artefacts.

In [van Harmelen et al. (2009)] an approach based on knowledge engineering has been applied to classify Semantic Web Applications with respect to reasoning patterns. In this work the objective is to abstract the tasks to design inference processes. Similarly, Semantic Web Applications have been classified in [Oren (2008)] by analysing the functionalities offered to the users with the aim of extracting generic requirements to support and improve the development methods. These efforts were targeted to understanding the structure of a system while we focus on the characterisation of the data manipulated in order to support the interpretation and documentation of its output.

[15] `http://rdf-vocabulary.ddialliance.org/discovery.html`.

In our work the goal is to devise possible relations between data objects from data flow descriptions, this can be a mean to extend the inference capabilities on workflow models (like provenance information, as we will show in Section 3.4).

Our generalisation can be used as a method to reduce a composite element of a workflow description (representing an event or an activity, for example) to a (set of) binary relation between the data items that have a role in it. In [Grewe (2010)] the author proposes a method to collapse an N-Ary relation to a set of binary relations trying to preserve the same semantic value. However, we do not intend to preserve the information of the activity in full, we just collapse it to express the relation between the nodes. Our model can leave out a number of entities in favour of making explicit the relations between data artefacts. An example of a similar simplification is in [Posada et al. (2005)], where authors reduce the expressiveness of a large dataset of CAD models in an ontology for the sake of supporting more effectively the task of reviewing in industrial design.

The approach of abstracting to optimise a given process can be comparable to the documentation of software architectures [Clements et al. (2002)]. The literature in this field emphasises the multiplicity of structures existing in information systems and how abstracting helps the understanding and reasoning over complex models.

3.6 Conclusions, perspectives and future work

Web Systems make use of a number of datasets with different scopes and relations between each other and external systems, covering aspects like acquisition, persistence, versioning, delivering, processing, distributing, and partitioning. Many of these operations include the usage of multiple data sets and target the creation of new ones from the sources. Understanding the propagation of the features of datasets from sources to derived datasets may be crucial if we want the data consumers to effectively benefit from them. Thus, reasoning on complex data flows becomes important and may be very complex, especially if relying on several different metalevel descriptions such as provenance, workflows, access control or terms of use. Being able to harmonise portions of different metalevel descriptions in a unified inference process can be beneficial in many contexts, as the scenario illustrated in Section 3.4. Datanode offers a foundational model to represent the dependencies between the data objects in a Web System. However, there might be many other metadata schemas that could be aligned and translated to datanode descriptions, covering scenarios that include data mining processes (described with NIF [Rizzo et al. (2012)], for example) or complex annotations (using EARMARK [Peroni and Vitali (2009)]). These applications still need to be explored and evaluated.

Datanode is a framework to design networks of data objects and to support deep analysis of the dependencies that they have in Web systems. Datanode does not offer an alternative to existing vocabularies but aims to be a method to organise knowledge in an unified way, so to foster the analysis and understanding of data collections on the Semantic Web. However, producing Datanode descriptions requires a deep understanding of the application (system or process). In Section 5.3 we propose an approach to support Data Cataloguers in the generation of Datanode annotation from existing workflow models.

In this Chapter, we concentrated on building a vocabulary of relations for the representation of processes as networks of data objects. The next step will be linking these relations to the second *knowledge component* for policy propagation: the licences associated with the data. In the next Chapter, we introduce the concept of Policy Propagation Rule (PPR) and observe the consequences of applying them to automatic reasoning at scale.

Chapter 4

Reasoning with Propagating Policies

Licences and data flows can be described through ontologies. The Open Digital Rights Language (ODRL)[1], introduced in Section 2.6, is an information model to support the exchange of policies on the World Wide Web, and can be used to represent permissions, prohibitions and duties within data Licences [Governatori et al. (2013a); Rodríguez-Doncel et al. (2014)]. The Datanode ontology[2], introduced in the previous Chapter, has been designed to formally describe how applications use data, their *data flows*, and consists in a taxonomy of ontological relations between data objects. Therefore, *processes* can be represented with Datanode as networks of data objects connected by semantic relations. The next step is to validate Hypothesis $H.\ 3$, and establish a semantic relation between types of data-to-data connections and policies. In particular, we operate a connection between Datanode relations and ODRL-described licences, so to determine whether a policy shall propagate or not. *Propagation rules* can be established in order to capture how policies, represented in ODRL, propagate in such data flows. However, having a description of policies and data flow steps implies a large number of rules to be specified and computed (number of policies times number of possible relations), raising the issue of effectively managing this rule base.

In this Chapter we aim at answering our second research question ($R.Q.\ 2$), and study how reasoning upon the propagation of policies can be practically performed. We introduce the concept of Policy Propagation Rule (PPR) and we establish a PPR rule base as a fundamental *knowledge component* for policy propagation, aside a catalogue of data licences expressed in ODRL, like the one developed in [Rodríguez-Doncel et al. (2014)]. On the one hand, we face the problem of producing and managing a PPR knowledge base. On the other hand, we develop a Policy Propagation Reasoner (PP Reasoner) and show the practicability of using it to reason on policy propagation, using the application scenarios described in Chapter 3.

[1] `https://www.w3.org/community/odrl/`

[2] `http://purl.org/datanode/ns/`

With the objective of developing and managing a PPR rule base, we introduce the (A)AAAA methodology, designed with the purpose of abstracting PPRs to reduce the number of rules to be managed. Moreover, we use this methodology to evolve Datanode to better fit the purpose of policy propagation. While the compression of the rule base reduces the number of rules to be managed, it requires the reasoner to compute more inferences. Therefore, we study the impact of rule base compression on the performance of the reasoning process. We practically evaluate a PP Reasoner, by performing a set of experiments with two different reasoning approaches: a) a Prolog based reasoner that produces inferences at query time; and b) an OWL reasoner in conjunction with SPIN rules that materialises inferences at load time.

This chapter is structured as follows. Section 4.1 presents an exemplary use case, and introduces the elements for reasoning on policy propagation, going through the description of the data flow, the representation of policies, and the concept of Policy Propagation Rule (PPR). Section 4.2 provides a detailed description of the (A)AAAA methodology, its application to evolve Datanode and develop consistent rules, and its evaluation as a method to compress the managed rule base. In Section 4.3, we report on experimental results about the impact of a compressed rule base on reasoning. For this purpose, we compare the performance of reasoning with an uncompressed rule base against reasoning with a compressed one. We perform this comparison using two different reasoners, the first computing the inferences at query time, the second materialising them at load time. Finally, we discuss our observations before closing the Chapter with some conclusions and perspectives on open issues.

4.1 Reasoning on policy propagation

In this section, we describe our approach for reasoning on policy propagation, and we present a use case as an example.

4.1.1 Approach

We define the problem of policy propagation as identifying the set of policies associated with the output of a process, implied by the policies associated with the input data source. In order to perform reasoning on policy propagation, we need:

a) descriptions of policies attached to data source,
b) a description of the data flow (the actions performed on the data), and
c) policy propagation rules (which actions do propagate a given policy).

Description of policies. We assume the policies of data sources are described as licenses or "terms and conditions" documents, and that they are expressed in RDF according to the ODRL ontology[3]. An ODRL `odrl:Policy` is an *entity to capture the statements of the policy*, specifying a set of `odrl:Rules`, each including a deontic aspect (`odrl:permission` or `odrl:prohibition`), which are defined for a set of `odrl:Actions` and a `odrl:target odrl:Asset`. Permissions, in turn, can comprise a `odrl:duty` (or more). For example, the RDF Licenses Dataset [Rodríguez-Doncel et al. (2014)] is a source of such descriptions. In our work, we also developed ad-hoc RDF documents to satisfy this requirement, when necessary.

Description of the data flow. Data flows are represented with the Datanode ontology, introduced in the previous Chapter. The terms are defined under the `http://purl.org/datanode/ns/` namespace (we use the prefix `dn:` for readability). The ontology defines a unique type - `dn:Datanode` - and 115 relations, starting from a single top property: `dn:relatedWith`, having `dn:Datanode` as `rdfs:domain` and `rdfs:range`. The relations are organized in a hierarchy by the means of `rdfs:subPropertyOf`. An instance of `dn:Datanode` is any data object that can be the input or output of a process.

Here we use the representations of data flows extracted from the descriptions of several Semantic Web applications prepared in the development of the ontology and reported in the previous Chapter.

Policy Propagation Rules. A Policy Propagation Rule (PPR) establishes a binding between a Datanode relation r and a policy p. A PPR is a Horn clause of the following form:

$$has(X,p) \wedge propagates(p,r) \wedge relation(r,X,Y) \rightarrow has(Y,p)$$

where X and Y are data objects, p is a policy and r is a Datanode relation between X and Y. When the policy p holds for a data object X, related to another data object Y by the relation r, then the policy p will also hold for the data object Y. For example a PPR could be used to represent the fact that downloading a file F distributed with an attribution requirement will result in a local copy D also needing to be used according to the attribution requirement. Therefore, the above abstract rule could be instantiated as follows:

$$has(F,attribution) \wedge propagates(attribution,isCopyOf) \wedge \\ relation(isCopyOf,F,D) \rightarrow has(D,attribution)$$

In fact, we can reduce a PPR to a more compact form, i.e. a binary association between a policy p and a relation r:

[3]ODRL 2.1: `https://www.w3.org/ns/odrl/2/ODRL21`.

$propagates(p, r)$

as the other components of the rule can be automatically derived for any possible X and Y having licences associated.

Table 4.1: Sources of Terms and conditions associated with the data sources of EventMedia[4].

Source	T&C
Flickr	Flickr APIs Terms of Use
Dbpedia	Creative Commons CC-BY-SA 3.0
Eventful	Eventful API Terms of Use
LastFM	LastFM Terms of Service
Upcoming	Non Commercial Use Requirement
Musicbrain	Creative Commons CC0
Foursquare	Foursquare Developers Policies

The proposed approach is based on reasoning with RDFS entailments (in particular the transitive property `rdfs:subPropertyOf`) in combination with a single generic rule. It is worth reminding that such a rule is a statement of the form *if* ... *then* ... that cannot be implemented in OWL2. The rule system we envisage here originates from a procedural rule (or production rule), whose meaning is *operational*, as it is derived from the execution of the actual symbol substitution [Hitzler et al. (2009)]. However, one problem with rules is the fact that they can lead to undecidability if they are used without restrictions. We base our discussion on the following considerations (taken from [Glimm et al. (2009)]):

- a rule is considered "safe" at the condition that only variables that occur in the body occur in the head.
- a rule is considered "safe" (DL-safe) if individual variables bind only to individuals named explicitly in the underlying knowledge base, and
- a rule is considered "safe" if it does not contain data values that are not explicitly mentioned in the underlying knowledge base. For example, a rule with a variable assuming any possible Integer values would generate an infinite number of statements.

[4]See also the following Web references:

Flickr APIs Terms of Use. `https://www.flickr.com/services/api/tos/`

Eventful API Terms of Use. `http://api.eventful.com/terms`

LastFM Terms of Service. `http://www.last.fm/api/tos`

Foursquare Developers Policies. `https://developer.foursquare.com/overview/community`

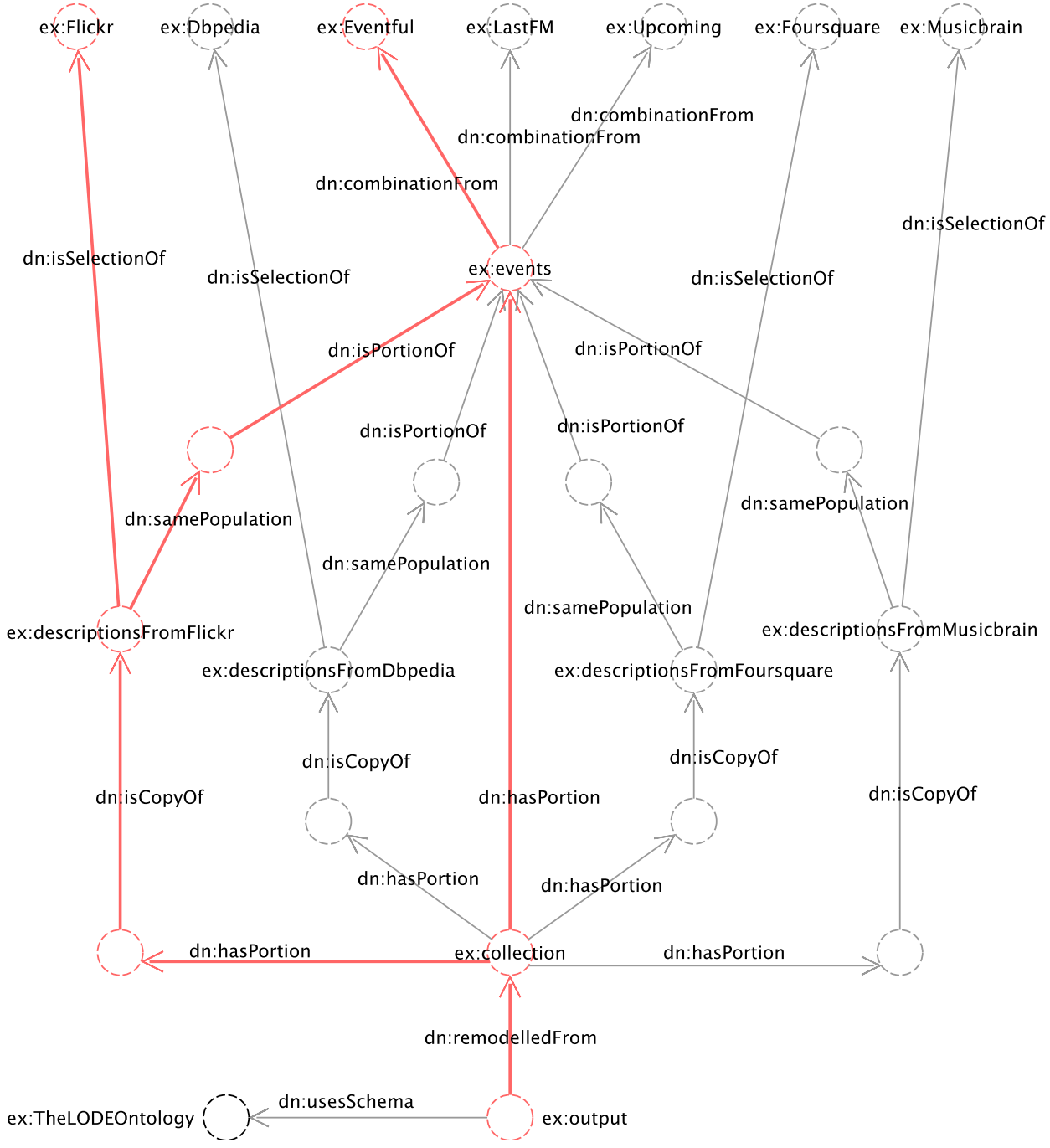


Figure 4.1: The data flow of EventMedia. Input sources are the top nodes. The node at the bottom depicts the output data, which is a remodelling of the data collected from various sources according to a specific schema.

It can be easily seen how our rule template satisfies all the above restrictions and therefore our inference system can be considered fully decidable.

One assumption we made in the introduction is that the output node must be a different artefact then the input ($A.\ 1$). However, this mechanism works well even with loops, as soon as they are represented as a single *flattened* iteration. In fact, since the policies of the output are always a subset of the ones of the input, it will be sufficient to execute the iteration one time in order to know the result. Any subsequent iteration would necessarily propagate all the remained policies.

With these elements established, we can trace the policies propagated within the data flow connecting input and output.

4.1.2 Example use case

We described the components required to reason upon policy propagation in data flows. We now introduce a motivating scenario. The following are the namespaces that will be referred to in this example:

```
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
odrl: < http://www.w3.org/ns/odrl/2/>
cc: <http://creativecommons.org/ns#>
dn: <http://purl.com/datanode/ns/>
ppr: <http://purl.com/datanode/ppr/ns/>
ex: <http://purl.org/datanode/ex/>
```

We selected EventMedia [Khrouf and Troncy (2012)] as an exemplary data-oriented system. EventMedia exploits real-time connections to enrich content describing events and associates it with media objects[5]. The application reuses data exposed by third parties, aggregating data about events and exposing them alongside multimedia objects retrieved on the Web. Aggregated data are internally represented using the LODE ontology [Shaw et al. (2009)]. In order to associate the right policies to these data, a description of the policies of the input data, a description of the data flow, and a knowledge base of PPRs are needed.

Table 4.1 lists the licenses or terms of use documents associated with the input data objects[6]. Listing 4.1 lists the set of policies associated to the content of the Flickr API, stated in the Flickr

[5]See `http://eventmedia.eurecom.fr/`.

[6]The *Upcoming* service is not available at the time of writing, however a snapshot of the documentation can be consulted from the Web Archive, reporting a non-commercial use clause: `https://web.archive.org/web/20130131064223/http://upcoming.yahoo.com/services/api/`. The application was firstly produced in 2014, when the EventMedia dataset description article was firstly submitted to the Semantic Web Journal. The description produced refers to the submitted version, which could be changed in the published version.

APIs Terms of Use[7].

Listing 4.1: Policies representation extracted from the Flickr APIs Terms of Use.

```
ex:FlickrTerms a odrl:Offer;
 rdfs:label "Flickr APIs Terms of Use";
 rdfs:seeAlso
  <https://www.flickr.com/services/api/tos/>;
 odrl:assigner <https://www.flickr.com> ;
 odrl:prohibition [
    odrl:target ex:Flickr ;
    odrl:action odrl:sell, odrl:grantUse,
            cc:CommercialUse ];
 odrl:permission [
    a odrl:Permission;
    odrl:target ex:Flickr ;
    odrl:action odrl:use;
    odrl:duty odrl:attribute ]
  .
```

Figure 4.1 illustrates the EventMedia data flow and Listing 4.2 the equivalent RDF description. Data are processed from event directories and enriched with additional information and media from sources like DBpedia[8], Flickr[9] or Foursquare[10]. In the figure, circles are data objects and arcs are Datanode relations. We will follow the path that connects the `ex:output` data object to two of the input data objects, namely `ex:Flickr` - that represents the Flickr API[11] (this path is highlighted in the figure), and Eventful[12] - a portal to search for upcoming events and related tickets. Apart from using the LODE ontology, `ex:output` is remodelled from an aggregation of various sources, named as `ex:collection`. The population (entities) of `ex:collection` includes `ex:events`, a `dn:combinationFrom` `ex:Eventful` with other sources (central path in the figure). Moreover, `ex:collection` includes descriptions of media from `ex:Flickr`, expressed by the path `dn:hasPortion` / `dn:isCopyOf` / `dn:isSelectionOf`. The data selected from `ex:Flickr` also refer to (some of) the entities aggregated in `ex:events`. This is expressed by the path `ex:descriptionsFromFlickr` `dn:samePopulation` / `dn:isPortionOf` `ex:events`. Therefore, the data flow is a backtrace of the abstract process of the EventMedia system, from the `ex:output` data object towards the input data sources.

[7]Flickr API Terms of Use: `https://www.flickr.com/services/api/tos/`.
[8]DBpedia: `http://dbpedia.org`.
[9]Flickr: `http://www.flickr.com`.
[10]This description has been initially elaborated in [Daga et al. (2014)].
[11]Flickr API: `https://www.flickr.com/services/api/`.
[12]Eventful: `http://eventful.com/`

Listing 4.2: The EventMedia data flow in RDF.

```
ex:events
    dn:combinationFrom ex:Eventful,
      ex:LastFM, ex:Upcoming .

ex:collection dn:hasPortion
 [dn:isCopyOf ex:descriptionsFromFlickr],
 [dn:isCopyOf ex:descriptionsFromDbpedia],
 [dn:isCopyOf ex:descriptionsFromMusicbrain],
 [dn:isCopyOf ex:descriptionsFromFoursquare],
 ex:events .

ex:descriptionsFromDbpedia
    dn:isSelectionOf ex:Dbpedia ;
    dn:samePopulation
        [ dn:isPortionOf ex:events ] .

ex:descriptionsFromFlickr
    dn:isSelectionOf ex:Flickr ;
    dn:samePopulation
        [ dn:isPortionOf ex:events ] .

ex:descriptionsFromFoursquare
    dn:isSelectionOf ex:Foursquare ;
    dn:samePopulation
        [ dn:isPortionOf ex:events ] .

ex:descriptionsFromMusicbrain
    dn:isSelectionOf ex:Musicbrain ;
    dn:samePopulation
        [ dn:isPortionOf ex:events ] .

:output
    dn:isRemodelledFrom ex:collection ;
    dn:usesSchema ex:TheLODEOntology .
```

The data flow described so far can be leveraged by a reasoner in conjunction with the ODRL policies of the inputs, and the PPRs, to infer the policies associated with `ex:output`. Listing 4.3 shows the policies propagated from the inputs to the output of the EventMedia data flow, some of them deriving from the restrictions applied to Flickr data, shown previously in Listing 4.1.

Listing 4.3: Example of policy associated with the output of EventMedia.

```
ex:outputPset a odrl:Set ;
  odrl:prohibition [
    odrl:target ex:output ;
    odrl:action odrl:modify,
        cc:commercialUse, odrl:sell ];
```

```
odrl:permission [
  odrl:target ex:output ;
  odrl:action odrl:use;
  odrl:duty odrl:attribute ]
.
```

4.2 (A)AAAA Methodology

Our approach considers the set of relations defined by Datanode and the policies defined in the RDF Licenses Database to generate a knowledge base of propagation rules. With the goal of improving the management of the rules, we study here to what extent it is possible to reduce the number of rules to be stored. This reduction requires to be complemented by inferences produced by a reasoner, relying on the axioms of the Datanode ontology. In the next Section we will assess whether this reasoning is practically feasible, and make the hypothesis that compressing the size of the rule base will not negatively impact the efficiency of the reasoner in computing the propagated policies.

Firstly introduced in [Daga et al. (2015a)], the (A)AAAA methodology covers all the phases necessary to set up a compact knowledge base of PPRs. The methodology is based on two assumptions: 1) Policy Propagation Rules are associations between policies and data flow steps, and 2) an ontology is available to organise data flow steps in a semantic hierarchy, e.g., for expressing the fact that relation *is a copy of* is a sub-relation of *is a derivation of*. In our work, we rely on Datanode as reference ontology, even if this is not required by the methodology itself. (The resulting compressed rule base will be the basis for our experiments in the remaining part of the Chapter.)

The methodology is composed of the following phases:

A1 **Acquisition.** The initial task is to set up a knowledge base of PPRs.

A2 **Analysis.** We apply Formal Concept Analysis (FCA). The FCA algorithm is performed on the knowledge base of PPRs. The output of the process is an ordered *lattice* of concepts: clusters of policies that propagate with the same set of relations.

A3 **Abstraction.** In this phase we search for matches between the ontology and the FCA lattice. When a match occurs, we subtract the rules that can be abstracted through the ontology's taxonomy.

A4 **Assessment.** We check to what extent a hierarchical organization of the relations matches the clusters produced by FCA (developing measures). This step

a) performs a *coherency check* between the lattice and the ontology (i.e. number of mismatches);

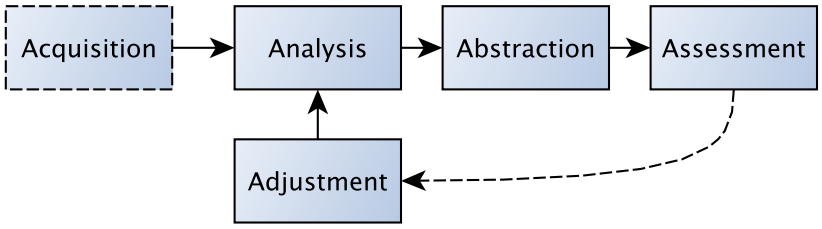


Figure 4.2: Phases of the (A)AAAA Methodology.

b) identifies all partial matches between clusters; and
c) evaluates how much the ontology compresses the knowledge base (i.e. the compression factor).

A5 **Adjustment.** Observing the measures produced in the previous phase, particularly about mismatches and partial matches, in this phase we perform operations changing the rule base or the ontology in order to repair mismatches, correct inaccuracies, evolve the ontology, and improve the compression factor as a consequence.

The (A)AAAA methodology constitutes an iterative process, as shown in Figure 4.2.

In the remaining part of this Section, we illustrate each phase of the methodology, by first describing the general approach and then explaining how it has been applied in practice.

4.2.1 Acquisition

The initial task is to set up a knowledge base of Policy Propagation Rules (PPRs). As presented in Section 4.1, a PPR can be conceived as an association between a policy and a possible relation between two data objects. The rule base can be represented as a binary matrix, where each row is a possible relation between two data objects, and each column a policy. The cells in the matrix can be 1 or 0, depending on whether the policy propagates given a particular relation. For example, the matrix might contain the cells

	duty shareAlike	permission reproduce	...
hasCopy	1	1	
hasDerivation	1	0	
...			

that can also be represented in CSV as follows:

```
hasCopy, duty shareAlike, 1
hasDerivation, duty shareAlike, 1
hasDerivation, permission reproduce, 0
```

From each positive cell in the matrix, we can generate a PPR, and populate the set of rules R.

This approach has been applied as follows. In order to setup the knowledge base of Policy Propagation Rules we relied on the RDF Licenses Database [Rodríguez-Doncel et al. (2014)], and extracted 113 possible policies. Each policy is an association of one deontic element (permission, prohibition or duty) and one action[13]. We used the Datanode ontology to extract a list of 115 possible relations between data objects. The combination of relations and policies lead to a matrix of 12995 cells. This phase required a manual supervision of all associations between policies and relations in order to establish the initial set of propagation rules, which has been performed with the support of the Contento tool [Daga et al. (2015b)]. At this stage, manual supervision was required to inspect all the cells and decide whether to establish a rule or not. The Contento tool allows us to incrementally inspect different portions of the context relying on filtering capabilities and bulk operations. We were able to check and uncheck collections of cells with similar status, and to keep track on the cells that were still to be supervised. Listing 4.4 displays a sample of binary relations that can be true or false in the matrix.

Listing 4.4: Example of cells of the binary matrix associating relations with policies

```
dn:hasPortion,permission odrl:copy,1
dn:identifiersOf,prohibition cc:DerivativeWorks,1
dn:isDependencyOf,permission odrl:derive,0
dn:usesSchema,permission cc:Reproduction,1
dn:processedInto,duty odrl:shareAlike,1
dn:isVocabularyOf,prohibition odrl:use,1
dn:metadata,prohibition odrl:transform,0
```

At the end of this process performed with Contento, the matrix had 3363 cells marked as true. The initial knowledge base was then composed of 3363 Policy Propagation Rules (Listing 4.5)[14].

Listing 4.5: Example of Policy Propagation Rules.

[13] We could have generated the policies by combining any ODRL action with any deontic component. However, this would have led to a large number of meaningless policies (eg: `duty odrl:use`). The adoption of the RDF Licence Database helped us to obtain a list of meaningful policies only.

[14] The reader can deduce that a large part of Datanode included relations that do not propagate any policy, for example the top relation `dn:relatedWith`, but also `dn:overlappingCapabilityWith`, `dn:about`.

$propagates(dn : hasPortion, permission\ odrl : copy)$
$propagates(dn : identifiersOf, prohibition\ cc : DerivativeWorks)$
$propagates(dn : usesSchema, permission\ cc : Reproduction)$
$ropagates(dn : processedInto, duty\ odrl : shareAlike)$
$propagates(dn : isVocabularyOf, prohibition\ odrl : use)$

4.2.2 Analysis

The objective of the second phase is to detect common behaviors of relations with respect to policy propagation. We achieve this by applying FCA, providing as input the binary matrix representation of the rule base R consisting of PPRs. The output of the FCA algorithm is an ordered set of *concepts* C. In FCA terms, each concept groups a set of objects (the concept's *extent*) and maps it to a set of attributes (the concept's *intent*). In our case, each concept maps a group of relations propagating a group of policies. These concepts are organized hierarchically in a *lattice*, ordered from the top concept T that includes all the objects and potentially no attributes, to the bottom concept B, including all the attributes with potentially an empty extent (set of objects). All other concepts are ordered from the top to the bottom. For example, usually a first layer of concepts right below T would include large groups of objects all having few attributes in common. Layers below would have more attributes and less objects, until the bottom B is reached. In our case, the top concept T would include all relations and no policies, while the bottom concept B includes all the policies but no relations. The concepts identified by FCA collect relations that have a common behavior in our rule base R, as they propagate the same policies.

Following this approach, we applied the FCA algorithm and obtained 80 concepts (in the first iteration of the methodology). Listing 4.6 shows one example. All the relations in the extent of this concept propagate the policies in the intent.

Listing 4.6: Example of a Concept.

```
Concept 71
Ancestors: 75
Descendants: 17,52,68,69
Extent [42]            Intent [9]
dn:cleanedInto         duty cc:Attribution
dn:combinedIn          duty cc:Copyleft
dn:duplicate           duty cc:Notice
dn:hasAddition         duty cc:ShareAlike
dn:hasAnonymized       duty cc:SourceCode
dn:hasAttributes       duty odrl:attachPolicy
dn:hasCache            duty odrl:attachSource
dn:hasChange           duty odrl:attribute
```

```
dn:hasComputation      duty odrl:shareAlike
dn:hasCopy
dn:hasDeletion
dn:hasDerivation
dn:hasExample
dn:hasExtraction
dn:hasIdentifiers
dn:hasInference
dn:hasInterpretation
dn:hasPart
dn:hasPortion
dn:hasReification
dn:hasSample
dn:hasSection
dn:hasSelection
dn:hasSnapshot
dn:hasStandIn
dn:hasStatistic
dn:hasSummarization
dn:hasTypes
dn:hasVocabulary
dn:identifiersOf
dn:isChangeOf
dn:isExampleOf
dn:isPartOf
dn:isPortionOf
dn:isSampleOf
dn:isSectionOf
dn:isSelectionOf
dn:isVocabularyOf
dn:optimizedInto
dn:processedInto
dn:refactoredInto
dn:remodelledTo
```

4.2.3 Abstraction

In this phase we apply a method for abstracting rules in order to reduce the size of the rule base. The abstraction process is based on applying an ontology that organises the relations in a hierarchy. For instance, the relation $hasCopy$ is a sub-relation of $hasDerivation$. Intuitively, a number of policies propagated by $hasDerivation$ should also be propagated by $hasCopy$ and all the other sub-relations in that branch of the hierarchy. By grouping all the relations below $hasDerivation$ in a transitive closure, we obtain a group of relations similar to the ones in the FCA concepts that we call the $hasDerivation$ *branch*. We expect branches of the ontology to be reflected in the clusters of relations obtained by FCA, thus we search for matches between the ontology and the FCA lattice. When a match occurs, we subtract the rules that can be abstracted.

Listing 4.7: Abstraction algorithm.

```
R = Rules()
C = FCAConcepts()
H = ComputeBranches()
ForEach (c,h) in (C,H)
        (pre,rec) = Match(c,h)
        when  pre == 1.0
            Subtract(R, Policies(c), Branch(h))
```

The process is summarised in Listing 4.7, and described as follows:

1. **Concepts.** From the result of the FCA algorithm we obtain a set of concepts C including relations r (extent of the concept) all propagating a set of common policies p (intent of the concept):

   $$C = ([r_1, p_1], [r_2, p_2], \ldots, [r_n, p_n])$$

2. **Branches.** For each relation in the ontology, we compute the transitive closure of its sub-relations, obtaining a set of mappings H:

   $$H = ([t_1, b_1], [t_2, b_2], \ldots, [t_n, b_n])$$

   where t is a relation in the ontology and b the related branch, i.e. all the sub-relations of t.
3. **Matching.** We search for (partial) overlaps between each branch h in H and relations in the extent of concept c of C (for simplicity, *in* c). The function $Match$ returns a measure of the matching between a branch and a concept, evaluated as *precision* and *recall*. In the case a branch is fully in the concept, we say the match has full precision (1.0). Inversely, recall indicates how much a branch covers the concept. A concept including only relations in the branch would result in a match with recall 1.0.
4. **Compression.** When a match has precision 1.0, we can use the relation h as abstraction for all the policies in the concept c. In other words, we can remove the rules referring to a subsumed relation, as they are implied by a more abstract one.

The result of the *Abstraction* phase can be represented as a set of measures between concepts and portions of the ontology. The measures we are interested in are:

– Extent size (ES). Number of relations in a concept.
– Intersection size (IS). Number of relations of the branch that are also present in the concept.
– Branch size (BS). Number of relations in the branch.

- Precision (Pre). It is calculated as $Pre = IS/BS$, meaning the degree of matching of a branch with the extent of the concept.
- Recall (Rec). Recall, calculated as $Rec = IS/ES$, i.e. how much the extent of the concept is covered by the branch.
- F-Measure (F1). F-Measure, the well-known fitness score, calculated from precision and recall as $F1 = 2 * ((Pre * Rec)/(Pre + Rec))$.

A general estimation of the effectiveness of the approach is given by the *compression factor* (CF). We calculate the CF as the number of abstracted rules divided by the total number of rules:

$$CF = \frac{|A|}{|R|}$$

with R the set of rules, and A the set of rules that can be abstracted.

This approach has been applied as follows. Listing 4.8 shows the output of the abstraction algorithm, returning the branches that are (partially) matched by a given concept.

For example, the extent of Concept 74 is a cluster of 43 relations, matching several branches in Datanode in different ways. At the beginning there is the Top Property of Datanode: `dn:relatedWith`. Its branch size (BS) has 115 relations, constituting the whole hierarchy in the ontology. Clearly, the size of the intersection is the same as the size of the extent (ES), as all the 43 relations of the concept are in the branch. However, the precision (pre) of the matching is pretty low: 0.37. This branch obviously matches the whole extent of Concept 74 with full recall (Rec) 1.0 - like with any other concept.

A more interesting case is the branch associated with `dn:hasPart`, whose intersection with the concept is over all 7 sub-relations. In fact, looking at the intent of Concept 74 in Listing 4.6, it sounds reasonable that all the parts of a given data object inherit all the cited duties[15]. The full precision enables the rules reduction process. A similar case is with the relation `dn:isVocabularyOf`.

Listing 4.8: Example of the matches between a concept and the branches in the Datanode hierarchy.

```
c    ES   IS   BS   Pre  Rec  F1   branch
74   43   43   115  0.37 1    0.54 dn:relatedWith
[...]
74   43   7    8    0.87 0.16 0.27 dn:sameCapabilityAs
74   43   7    7    1    0.16 0.28 dn:hasPart
74   43   6    7    0.86 0.14 0.24 dn:isPartOf
74   43   3    6    0.5  0.07 0.12 dn:hasVocabulary
74   43   6    6    1    0.14 0.25 dn:isVocabularyOf
```

[15]They inherit many other policies as well, and those are considered by other concepts in a lower layer of the FCA lattice.

```
[...]
```

By only considering the branches matched with full precision by each concept, we can subtract 1925 rules, for a compression factor CF of 0.572, in the first iteration of our methodology.

It is worth noting that this process only aims to identify matches between the rule base and the subsumption hierarchy of the ontology, and does not account for mismatches. In principle, it is possible that some policies propagated by a relation are not propagated by one of its sub-relations. In this situation, a reasoner that want to benefit from a compressed rule base will return also wrong results. We deal with this problem in the *Assessment* phase.

4.2.4 Assessment

The objective of this phase is to assess to what extent the ontology and the FCA lattice are coherent. In particular, we want:

1. to detect mismatches (*coherency check*) to be resolved before using the compressed rule base with a reasoner, and
2. to identify *quasi matches* that could be boosted to become a full match performing changes in the rule base or the ontology

Coherency check. The abstraction process is based on the assumption that it is possible to replace asserted rules with inferences implied by subsumed relations in the ontology. This implies that all policies propagated by a given relation must be propagated by all sub-relations in the original (uncompressed) rule base. A coherency check process is necessary to identify whether this assumption does hold for all the relations in the extent. In case it does not, we want to collect and report all these mismatches in order to be able to fix them at a later stage in the methodology. Listing 4.9 shows the algorithm used to detect such problems on a given concept in the lattice.

Listing 4.9: Coherency check algorithm.

```
c =                          // Given a concept
M = []                       // A list to record mismatches
S=SuperConcepts(c)           // Get the set of super concepts
ForEach s in S               // For each super concept
 E=Extent(c)                 // Get the relations in the concept
 E1=Extent(s)                // Get the relations in the super concept
 ForEach(e1 in E1)           // If any relation in the super concept (e1)
  if not(Contains(E,e1))     //  is not in the extent of the concept, and
   ForEach e in E            //  if the missing relation is in the branch
```

```
    if Contains(Branch(e),e1) //  of any relation in the concept
     M = [e,e1]|M             // Append the pair of relations to the list
return M
```

We know from the definition of a FCA lattice that super-concepts will include a larger set of relations propagating a smaller number of policies. Given a concept c, the algorithm extracts the relations (extent) of each of any superconcept (S denotes the set of all super concepts s of c). In case these relations are not also present in (the extent of) c, it is mandatory that they are not sub-relations of any relation in the extent of c. In case they are, this means that a sub-relation is not inheriting all the policies of the parent one, thus invalidating our assumption. Mismatches M are identified and reported. Listing 4.10 shows the result of the algorithm for a concept. In this example, a number of sub-relations of `dn:isVocabularyOf` do not propagate some of the policies of Concept 71.

Listing 4.10: Coherency check result for Concept 71: mismatches.

```
Concept  Branch              Relation
71       dn:isVocabularyOf  dn:attributesOf
71       dn:isVocabularyOf  dn:datatypesOf
71       dn:isVocabularyOf  dn:descriptorsOf
71       dn:hasVocabulary   dn:hasDatatypes
71       dn:hasVocabulary   dn:hasDescriptors
71       dn:hasVocabulary   dn:hasRelations
71       dn:isChangeOf      dn:isAdditionOf
71       dn:isChangeOf      dn:isDeletionOf
71       dn:isVocabularyOf  dn:relationsOf
71       dn:isVocabularyOf  dn:typesOf
```

Quasi matches. The result of the *Abstraction* phase includes a set of measures between concepts and portions of the ontology. Table 4.2 shows an example of the measures obtained applying the approach in the context of the Datanode ontology. The measures defined in the *Abstraction* phase are now considered to quantify and qualify the way the ontology aligns with the propagation rules: precision and recall indicate how much a relation is close to being a suitable abstraction for policy propagation. Some general considerations can be made by inspecting these measures. When $Rec = 1$, the whole extent of the concept is in the branch. The branch might also include other relations, which do not propagate the policies included in the concept. When $Pre = 1$, we can perform the abstraction of rules, as in Listing 4.7. A low recall indicates that a high number of exceptions still need to be kept in the rule set. It also reflects a high ES, from which we can deduce a low number of policies in the concept. As a consequence of that, inspecting a partial match with high precision and low recall highlights a problem that might be easy to fix, as the number of

Table 4.2: Excerpt from the table of measures computed by the abstraction algorithm in Listing 4.7. c=Concept ID, ES=Extent Size, IS=Intersection Size, BS=Branch size, Pre=Precision, Rec=Recall, F1=F-Measure.

c	ES	IS	BS	Pre	Rec	F1	Branch
79	52	52	115	0.45	1	0.62	relatedWith
77	46	19	21	0.9	0.41	0.56	hasDerivation
75	44	8	11	0.73	0.18	0.29	samePopulationAs
67	35	7	7	1	0.2	0.33	hasPart
67	35	6	7	0.86	0.17	0.28	isPartOf
36	16	3	3	1	0.19	0.32	hasCopy
36	16	3	3	1	0.19	0.32	isCopyOf
24	12	6	6	1	0.5	0.67	hasVocabulary
9	8	1	1	1	0.12	0.21	hasReification
0	4	4	115	0.03	1	0.06	relatedWith

relations and policies to compare will be low. For example, row 2 of Table 4.2 relates to a relation with $BS - IS = 2$, so we need only to check whether 2 relations in the `hasDerivation` branch might also propagate the policies in concept 77. The perfect match between a concept and a branch of the ontology would be $F1 = 1$. However, when this does not happen we can try to improve the approximation.

At this stage we can make the following considerations:

- The presence of mismatches between the lattice and the ontology will make the reasoner return wrong results, thus they must be eliminated.
- The size of the matrix that was manually prepared in the *Acquisition* phase is large ($\tilde{1}$3k cells), and even with the support of the Contento tool it is still possible that errors or misjudgments have been made at that stage of the process.
- The Datanode ontology has not been designed for the purpose of representing a common behavior of relations in terms of propagation of policies. It is probably possible to refine the ontology in order to make it cover this use case better (and reduce the number of rules even more).

4.2.5 Adjustment

In this phase, we try to make adjustments in order to (a) repair mismatches identified in the assessment phase (b) evolve the rule base R (adding or removing rules), and (c) modify the ontology hierarchy H, in order to improve the compression factor CF. Six operations can be performed: *Fill*, *Wedge*, *Merge*, *Group*, *Remove*, *Add*. Here we briefly illustrate the property of each operation, referring to [Daga et al. (2015a)].

Fill.

Sometimes a branch is not completely in the concept. We can inspect the missing relations, and realise that they should all be in the concept (diagnosis). This operation is illustrated by Figure 4.3. The ticked circles represents relations in the ontology that are in the extent of a given concept. Unticked circles are missing, meaning that they do not propagate the policies in the concept. The Fill operation makes a branch b be fully in concept c, attempting to push Pre up to 1. This is achieved by adding to R all the rules generated from the association between the policies in concept c and the relations in branch b. This change affects the PPR knowledge base R, increasing the number of rules.

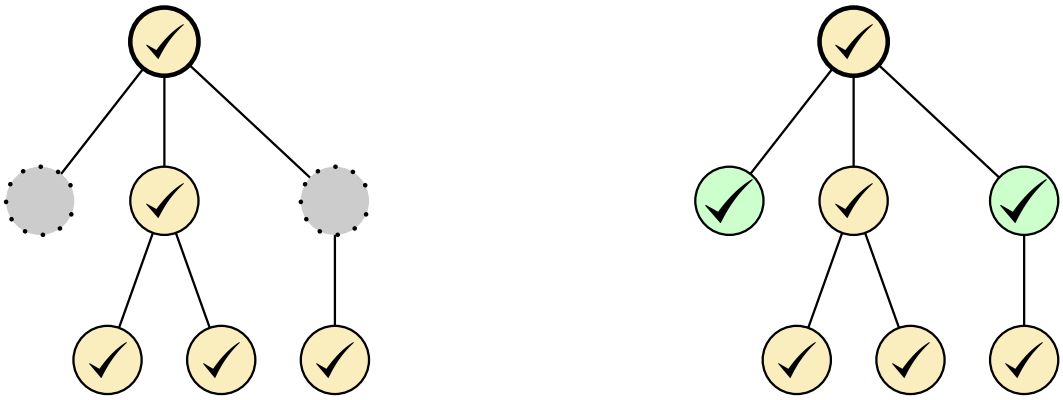


Figure 4.3: The *Fill* operation. The diagram on the left shows the diagnosis of the issue, while the diagram on the right shows the way it was repaired.

Wedge

Sometimes a branch is abstracted by a relation that is too general, so that all its sub-relations actually propagate the policies in c but the top relation cannot. Figure 4.4 illustrates the operation. As a

result, a new relation is wedged between a top relation and all its direct subproperties. The new branch will allow us to perform a *Fill* operation, in the next iteration.

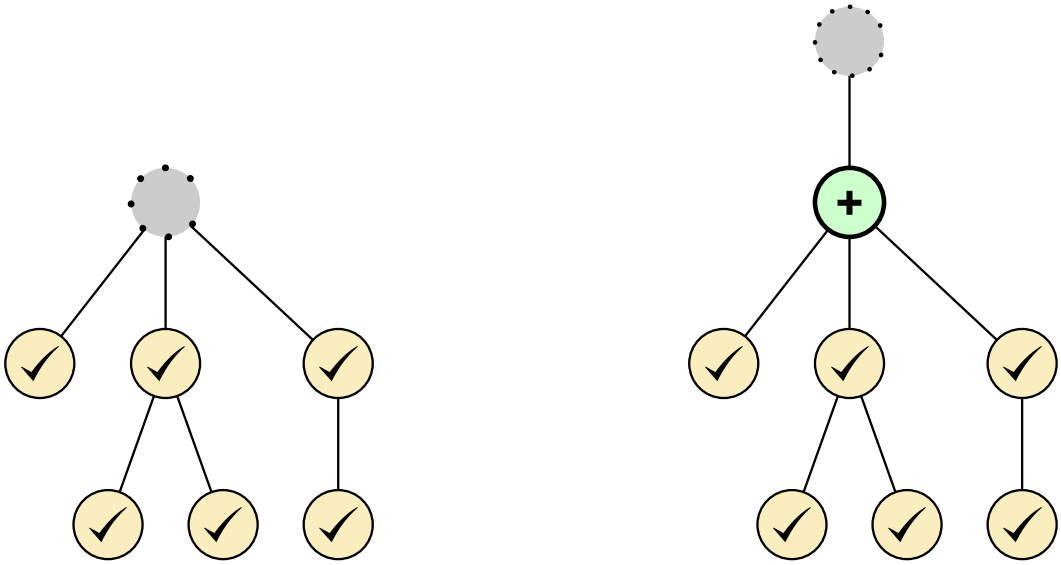


Figure 4.4: The Wedge operation.

Merge

We observe a concept matched by two branches with maximum precision. It is possible that two top relations can be abstracted by a new common relation. If this is the case we perform this operation and create a new relation, as shown in Figure 4.5. The new branch will allow us to perform a *Fill* operation in the next iteration.

Group

Figure 4.6 illustrates the *Group* operation. A set of relations are all together in the extent of a concept, but belong to different branches. We want to create a common parent relation for them. Again, the new branch enables a *Fill* operation, to be executed in the next iteration.

Remove and Add

We can *Remove* a relation as a sub-property of another (and possibly cut a sub-branch). This operation removes a single subsumption relation in the ontology. For example, a relation is not in a concept, the branch is almost all there, and we realise that the missing relation should not actually

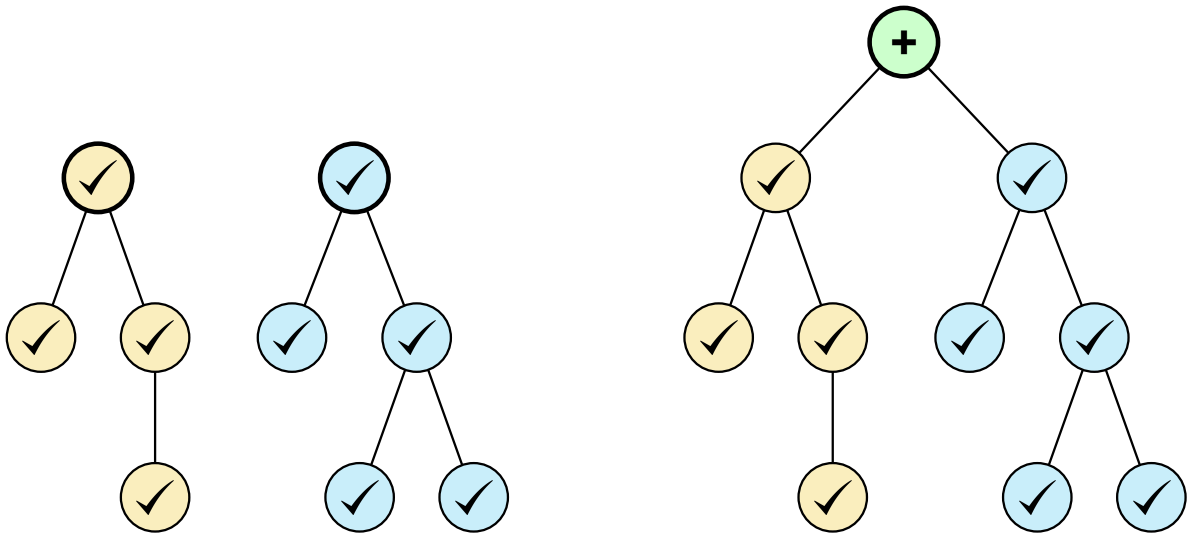


Figure 4.5: The Merge operation.

be part of it. Basically we detail the semantics of the relation and we remove it from the branch. After that, we might relocate it elsewhere with a *Add* operation.

After a change to the ontology hierarchy (for example the addition of a new relation), normally the *Fill* operation is performed on the newly created branch, in order to populate the rule base accordingly. Except for the *Fill* operation, all the operations are performed on the ontology (Datanode, in our case). As shown in Figure 4.2, after the *Adjustment* phase we restart a new iteration.

In what follows we illustrate three examples of changes performed during our application of the methodology.

Example 1.

The *Assessment* phase of the methodology reported possible mismatches between the FCA output and the ontology hierarchy. These errors must be repaired if we want the compressed rule base to be used by a reasoner. For example, Listing 4.10 shows the set of mismatches detected for concept (71). In this list, the `dn:isVocabularyOf` branch contains a number of relations that do not propagate the related policies, breaking the assumption that all the policies of `dn:isVocabularyOf` are also propagated by all the other relations in his branch. With a *Fill* operation, we can add all the necessary rules to remove this mismatch.

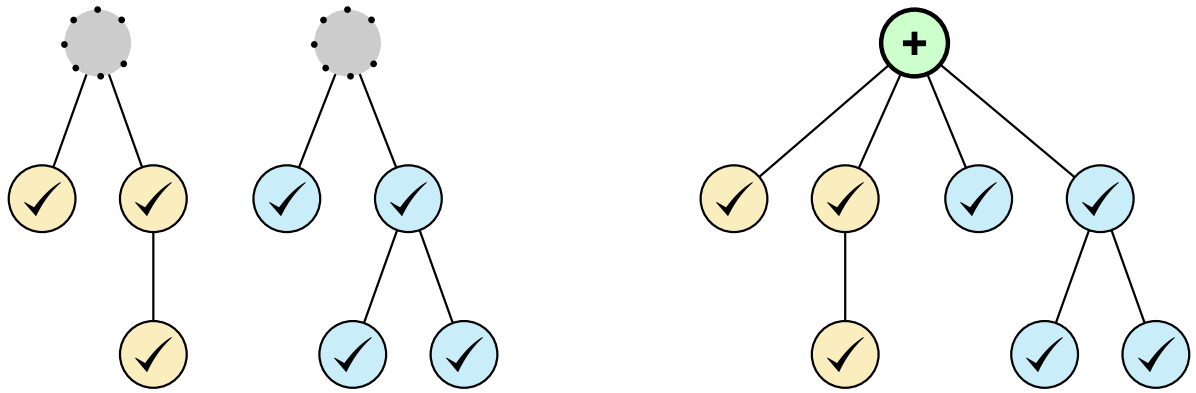


Figure 4.6: The Group operation.

Example 2.

In Section 4.2.4 we described a method to catch possible errors in the rule base, based on the identification of partial matches with high precision and low recall. Such cases highlight a branch that is close to be fully included in a concept. As example, we can pick branch `dn:isPartOf` from Listing 4.8. Listing 4.11 shows the details of how the concept matches this branch. It happens that all relations except `dn:isSelectionOf` are part of this concept. In other words, they propagate the policies listed in the intent of Concept 74 (Listing 4.6). However, this is a mistake that happened during the *Acquisition* phase, as `dn:isSelectionOf` should behave in a similar way to `dn:isExampleOf`.

Listing 4.11: Example of the matches between a concept and the branches in the Datanode hierarchy.

```
c    es   is   bs   pre  rec  f1   branch
74   43   6    7    0.86 0.14 0.24 dn:isPartOf
                         + dn:isPartOf
                         ! dn:isSelectionOf
                         + dn:isExampleOf
                         + dn:isSectionOf
                         + dn:identifiersOf
                         + dn:isPortionOf
                         + dn:isSampleOf
```

We decide then to perform a *Fill* operation, adding all the necessary rules to make the branch `dn:isPartOf` fully covering the intent of Concept 74.

Example 3. The `dn:sameCapabilityAs` branch seems to have high precision and low recall as in the previous example. Listing 4.12 shows that the only missing relation is the top one. In Datanode, `dn:sameCapabilityAs` is defined as the relation between two objects having the same vocabulary and the same population (containing actually the same data). However, it is possible that two objects have the same "data" without having the same policies. For example, datasets like lists of cities or postcodes might be imported from different sources, and may have different policies while containing the same data! In this case we opted for adding a new relation to Datanode that can abstract all the branches with precise semantics: `dn:sameIdentityAs`. `dn:sameIdentityAs` tries to capture exactly the fact that two data objects share the same origin, the same population and the same vocabulary. The operation performed to add this relation is *Wedge*, as the new property is injected between `dn:sameCapabilityAs` and its direct sub-relations.

Listing 4.12: Example of the matches between a concept and a branch in the Datanode hierarchy.

```
c    es   is   bs   pre  rec  f1   branch
74   43   7    8    0.87 0.16 0.27 dn:sameCapabilityAs
                          ! dn:sameCapabilityAs
                          + dn:hasCopy
                          + dn:hasSnapshot
                          + dn:hasCache
                          + dn:isCopyOf
                          + dn:isSnapshotOf
                          + dn:isCacheOf
                          + dn:duplicate
```

After each operation we run our process again from the *Analysis* phase to the *Assessment*, in order to evaluate whether the change fixed the mismatch and/or how much the change affected the compression factor. The process is repeated until all mismatches have been fixed, and there are no others quasi-matches that can be adjusted to become full matches. Moreover, when new policies are defined in the database of licenses (the RDF licence Database, in our work), the process has to be repeated in order to insert the new propagation rules. However, this is only required after changes in the licenses, as changes in the associations between policies and data objects do not affect the PPRs, e.g., changing the licence of a data source.

4.2.6 Evaluation of the CF

In the previous sections we described the phases of the (A)AAAA methodology, and how we applied it to the task at hand. Figure 4.7 shows how the compression factor CF increases with the number

of adjustments performed, while Figure 4.8 illustrates the progressive reduction of mismatches. Details about the changes performed are provided in Table 4.3 (identified by the symbol $+$), which also includes statistics about number of mismatches ($\neq$), the impact on number of rules (R), number of concepts generated by FCA (C), number of rules abstracted (A), remaining rules (R_+), and compression factor (CF).

Thanks to this methodology we have been able to fix many errors in the initial data, to refine Datanode by clarifying the semantics of many properties and adding new useful ones. As final result we obtained: 4225 rules in total, 34 concepts, 3451 rules abstracted and 774 rules remaining, boosting the CF up to **0.817**.

The version of the ontology prior to peforming such changes can be found at `http://purl.org/datanode/0.3/ns/` and the modified version of the ontology can be found at `http://purl.org/datanode/0.5/ns/`. As previously mentioned, the *Acquisition* phase has been performed with the Contento tool [Daga et al. (2015b)]. The tools used in the other phases of the methodology, from the *Abstraction* to the *Adjustment* phases, can be found at `https://github.com/enridaga/ppr-a-five`.

4.3 Experiments

The methodology described in the previous section allows us to reduce the number of rules that need to be stored and managed. The results of applying this methodology on the PPR knowledge base derived from the RDF Licenses Dataset, show how the compression factor can be dramatically increased after several iterations. Our assumption in this work is that it might positively affect the performance of reasoning on policy propagation. Here, we therefore assess through realistic cases the performance of reasoners when dealing with a compressed knowledge base of PPRs, as compared to when dealing with the uncompressed set.

We took 15 data flow descriptions from the ones developed in Chapter 3, referring to Semantic Web applications that rely on data obtained from the Web. Each data flow represents a data manipulation process, consuming a data source (sometimes multiple sources), and returning an output data object. Given a set of policies P_i associated with the input data, the objective of a reasoner is to find the policies P_o associated with the output of the data flow. The experiments have the objective to compare the performance of a reasoner when using an uncompressed or a compressed rule base respectively. Therefore, each reasoning task is performed twice: a first time, to provide the full knowledge base of PPRs; the second time, to provide the compressed knowledge base in conjunction with the hierarchy of relations of the Datanode ontology (required to produce

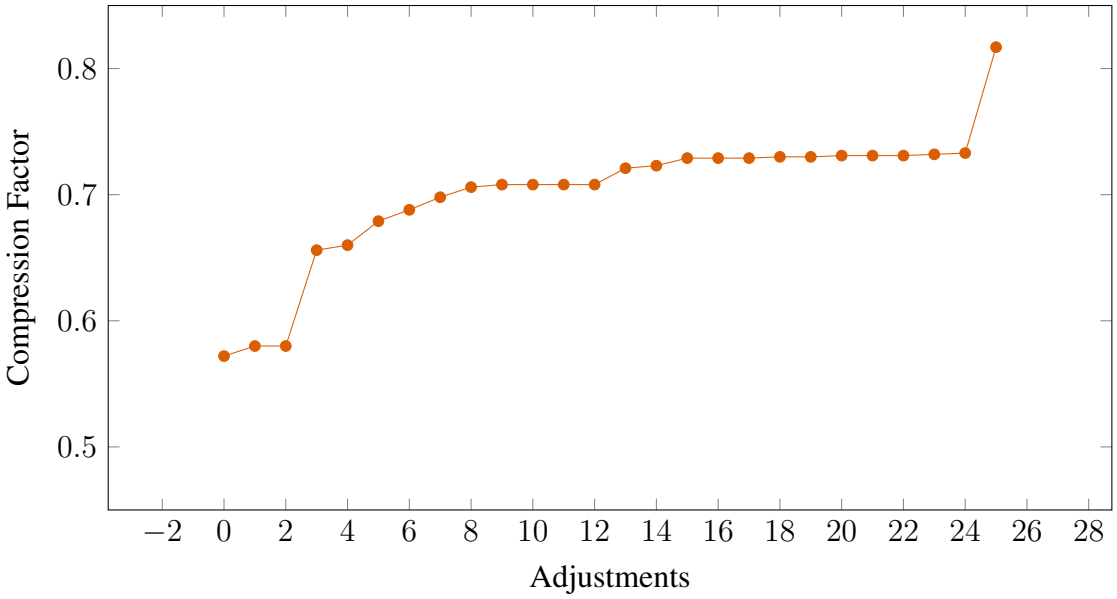


Figure 4.7: Progress of the Compression Factor (CF).

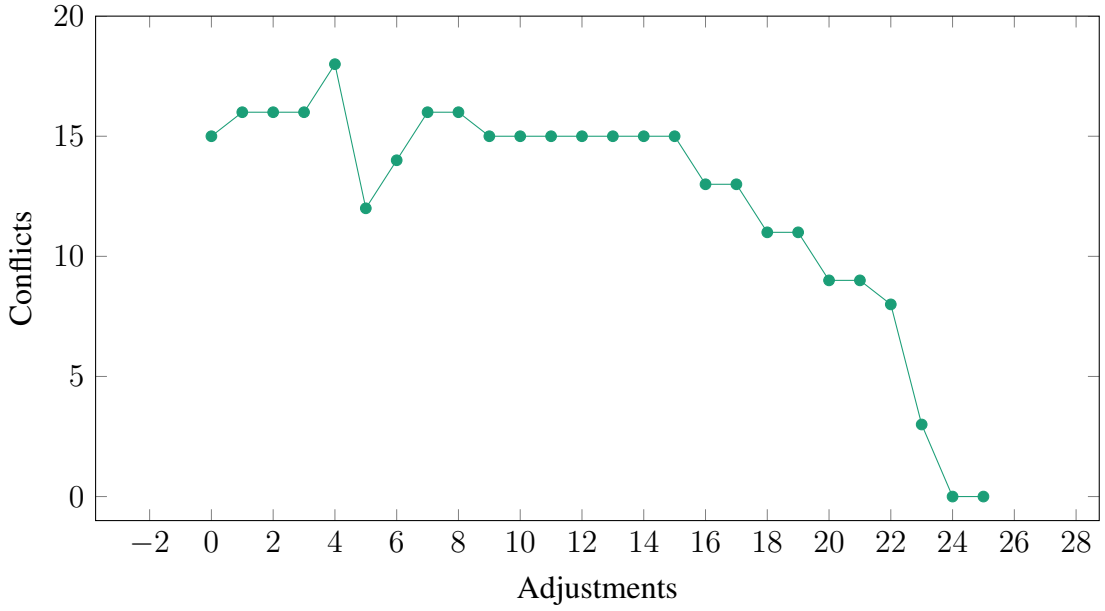


Figure 4.8: Progress in the number of conflicts.

Table 4.3: List of changes performed.

+	C	$\neq$	R	A	R_+	CF
0	80	15	3363	1925	1438	0.572
1	80	16	3370	1953	1417	0.58
2	80	16	3370	1953	1417	0.58
3	80	16	3480	2283	1197	0.656
4	80	18	3482	2299	1183	0.66
5	78	12	3500	2376	1124	0.679
6	78	14	3608	2484	1124	0.688
7	78	16	3716	2592	1124	0.698
8	96	16	3822	2698	1124	0.706
9	93	15	3824	2706	1118	0.708
10	93	15	3824	2706	1118	0.708
11	93	15	3824	2706	1118	0.708
12	93	15	3824	2706	1118	0.708
13	76	15	3837	2765	1072	0.721
14	76	15	3844	2778	1066	0.723
15	78	15	3865	2817	1048	0.729
16	78	13	3866	2818	1048	0.729
17	78	13	3874	2826	1048	0.729
18	63	11	3878	2830	1048	0.73
19	63	11	3882	2834	1048	0.73
20	63	9	3892	2844	1048	0.731
21	55	9	3897	2849	1048	0.731
22	60	8	3898	2850	1048	0.731
23	60	3	3908	2860	1048	0.732
24	54	0	3914	2870	1044	0.733
26	34	0	4225	3451	774	**0.817**

The first column identifies the change performed (starting from the initial state).

C = Number of concepts in the FCA lattice

$\neq$ = Number of mismatches between the FCA lattice and the ontology

R = Number of rules before the process

A = Number of rules abstracted (subtracted)

R_+ = Size of the compressed rule base (without the abstracted rules)

CF = Compression Factor

Table 4.4: Data flows used in the experiments.

Data flow	has policy	has relation	relations	data objects	policies	sources	output policies
AEMOO-1	6	18	13	10	6	1	6
DBREC-1	6	2	2	2	6	1	3
DBREC-2	6	8	7	5	6	1	6
DBREC-3	6	12	7	8	6	1	6
DBREC-4	6	14	8	9	6	1	3
DBREC-5	6	14	10	10	6	1	6
DBREC-6	6	10	10	6	6	1	3
DBREC-7	6	9	6	10	6	1	6
DBREC-8	6	5	4	5	6	1	6
DISCOU-1	7	22	10	14	7	1	5
DISCOU-11	5	13	9	12	5	1	0
EventMedia-1	37	25	8	24	25	6	4
REXPLORE-1	16	14	8	14	8	3	3
REXPLORE-2	32	23	4	18	14	6	6
REXPLORE-4	32	18	8	14	15	6	3

Highlighted are the maximum and minimum values for each of the data flow inputs. In one case (DISCOU-11), none of the policies attached to the source are propagated to the output.

the inferences).

Reasoners infer logical consequences from a set of asserted facts and inference rules (knowledge base). A reasoner can compute the possible inferences from the rules and the facts any time it is queried, thus exploring the inferences required to provide the complete answer. Alternatively, a reasoner can compute all possible inferences at the time the knowledge base is loaded, and only explore the materialised facts at query time. In order to appropriately address both of those reasoning strategies, we run the experiments with two different reasoners. The first reasoner performs the inference at query time using a backward chaining approach; is implemented as Prolog program and we will refer to it as the *Prolog* reasoner. The second reasoner computes all the inferences at loading time (materialisation); is implemented as an RDFS reasoner in conjunction with SPIN rules, and we will refer to it as the *SPIN* reasoner. Both reasoners are implemented in Java within the PPR Reasoner project[16]. Both reasoners have the capability of executing PPRs and expand the results according to the approach presented in the previous sections.

The *Prolog* implementation is a program relying on JLog, a Prolog interpreter written in Java[17]. The program incorporates a *meta rule* that traverses the set of PPRs, encoded as facts. At the same time, it supports the subsumption between relations. Listing 4.13 shows an excerpt of the program.

Listing 4.13: Excerpt of the Prolog reasoner program.

```
i_rdfs_sub_property_of(X,X) .
i_rdfs_sub_property_of(X,Y) :-
       rdfs_sub_property_of(X,Z),
       i_rdfs_sub_property_of(Z,Y) .
i_propagates(X,Y) :- propagates(X,Y) .
i_propagates(X,Y) :-
       i_rdfs_sub_property_of(X,Z),
       propagates(Z,Y) .
i_has_policy(T,P,_) :- has_policy(T,P) .
i_has_policy(T,P,L) :-
       i_has_relation(S,T,R),
       not(visited(S,L)),
       i_propagates(R,P),
       i_has_policy(S,P,[S|L]) .
i_has_policy(T,P) :- i_has_policy(T,P,[]) .
```

The *SPIN* reasoner is built upon the RDFS reasoner of Apache Jena[18] in combination with the

[16]PPR Reasoner: `https://github.com/enridaga/pprreasoner`. The experiments were performed within the *ppr-evaluation* module, which includes instructions about how to reproduce them.

[17]`http://jlogic.sourceforge.net/`

[18]`http://jena.apache.org/`

SPIN engine[19], a tool that allows developers to define rules using SPARQL. The core part of the reasoner executes PPRs as a SPARQL meta query (Listing 4.14).

Listing 4.14: Construct meta-query of the *SPIN* reasoner.

```
CONSTRUCT {
  ?this ppr:policy ?policy
} WHERE {
  ?int ?relatedWith ?this .
  ?int ppr:policy ?policy .
  ?relatedWith ppr:propagates ?policy
}
```

We performed the experiments with the data flows listed in Table 4.4. Each data flow describes a process executed within one of the 5 systems selected as exemplary data-oriented applications. These data flows were formalised at a previous stage with the purpose of testing the Datanode ontology, and were reused for the experiments without changes. However, information about the policies of the input was added. Table 4.4 illustrates the properties of these data flows, and compares them along several dimensions. The *has policy* column reports the number of statements about policies, from a minimum of 5 to 37 policies. The size of the data flow is reported in the *has relation* column of the table, as it is measured in number of Datanode relations used, spanning from 2 to the maximum of 25. The *relations* column reports the number of distinct relations, the same applying to *data objects*, *policies*, *sources* and the propagated *output policies*. Highlighted are the maximum and minimum values for each of the dimensions. In one case (DISCOU-11), none of the policies attached to the source are propagated to the output.

Each experiment takes the following arguments:

– *Input*: a data flow description
– *Compression*: $True/False$
– *Output*: the output resource to be queried for policies

In case *compression* is $False$, we provide the complete knowledge base of PPRs as input of the reasoning process without including information on subsumption between the relations described in the dataflow. Conversely, when *compression* is set to $True$, the compressed PPR knowledge base is used in conjunction with the Datanode ontology. It is worth noting that the (A)AAAA methodology is also an ontology evolution method, as most of the operations targeted to improve the compression of the rule base are performed on the ontology by adding, removing and replacing relations in the

[19] http://spinrdf.org/

hierarchy. In these experiments, we are considering the evolved rule base (and ontology), which has been harmonised by fixing mismatches between the rule set and the ontology.

The experiments were executed on a MacBook Pro with an Intel Core i7/3 GHz Dual Core processor and 16 GB of RAM. In case a process was not completed within five minutes, it was interrupted. Each process was monitored and information about CPU usage and RAM (RSS memory) was registered at intervals of half a second. When terminating, the experiment output would include: total time (t), resources load time (l), setup time (s), and query time (q). The size of the input for each experiment is reported in the diagrams in Figure 4.9.

We consider performance on two main dimensions: time and space.

Time performance is measured under the following dimensions:

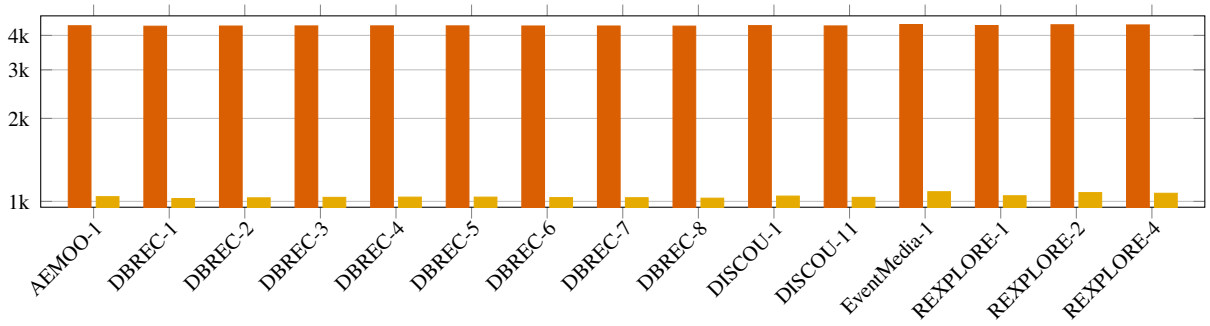
- L Resources load time.
- S Setup time. It includes L, in addition to any other operation performed before being ready to receive queries (e.g., materialization).
- Q Query time.
- T Total duration: $T = S + Q$.

Space is measured as follows:

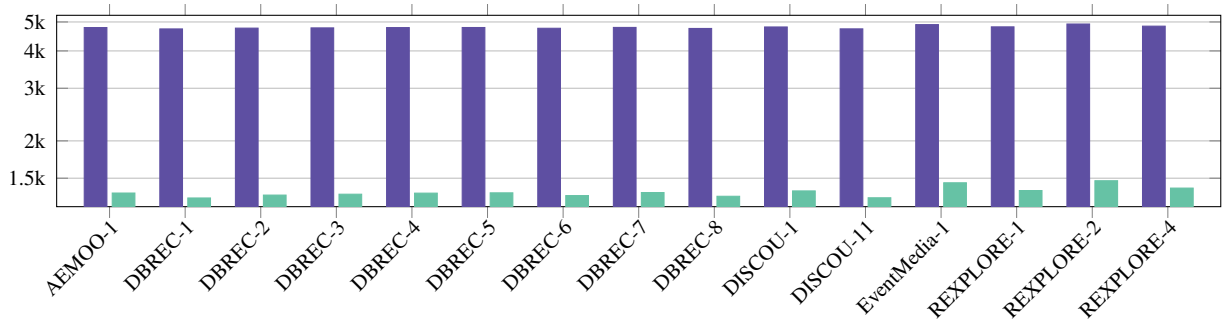
- Pa Average *CPU* usage.
- M Maximum memory required by the process

Each experiment was executed 20 times. We compared the results of the experiments with and without compression, and verified they included the same policies. In the present report, we show the average of the measures obtained in the different executions. In order to evaluate the accuracy of the computed average measure from the twenty executions of the same experiment, we calculated the related *Coefficient of Variation* (CV)[20]. CV is a measure of spread that indicates the amount of variability relative to the mean. A high CV indicates a large difference in the observed measures, thus reducing the significance of the computed mean. Diagrams 4.10a and 4.10b display the CV of all the measures for the *Prolog* and *SPIN* reasoner, respectively. In almost all the cases the CV for the *Prolog* reasoner was below 0.1, with the exception of memory usage M, which in many cases showed a fluctuation between 0.2 and 0.4. Experiments with the *SPIN* reasoner reported a much more stable behaviour in terms of consumed resources, the CV being assessed below 0.1 in almost all the cases, except the Query time of some experiments (the peak is on DBREC-4). However, Q with the *SPIN* reasoner were fluctuating around an average of 10ms, making the observed variation

[20]Coefficient of Variation, also known as Relative Standard Deviation (RSD). `https://en.wikipedia.org/wiki/Coefficient_of_variation`

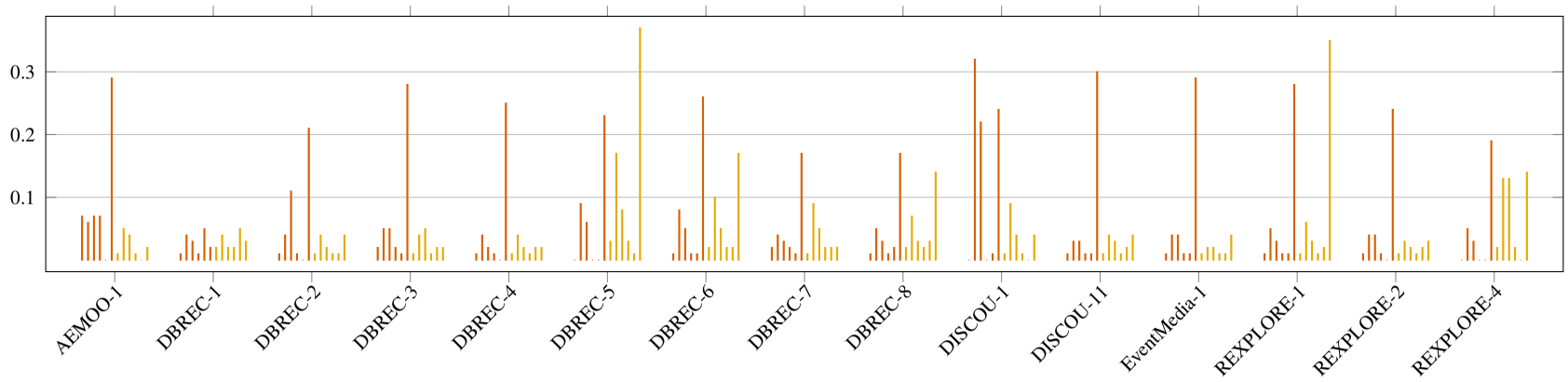


(a) *Prolog* reasoner: input size computed as number of Prolog facts with the original (dark orange) and compressed (light yellow) input for each data flow.

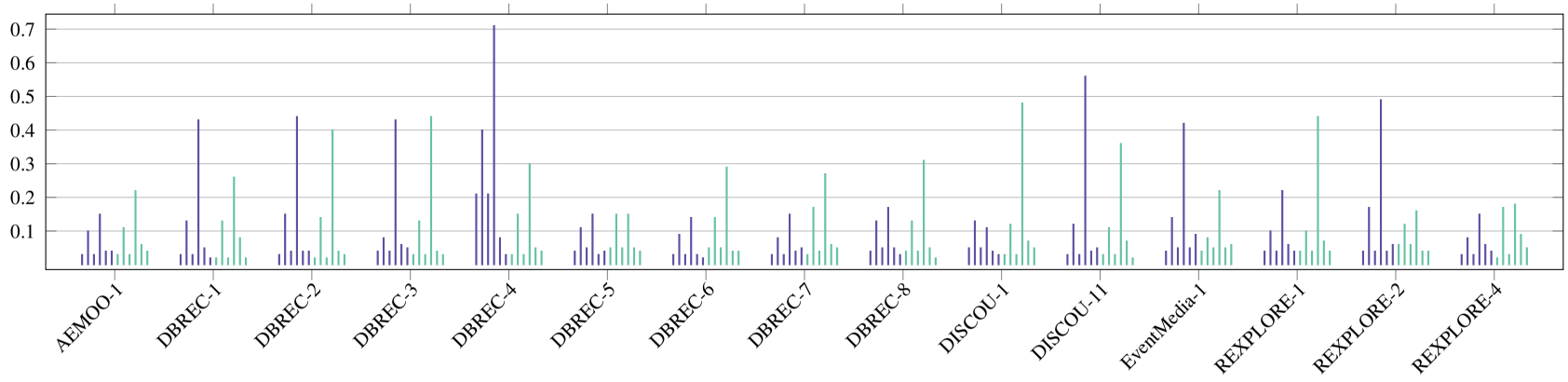


(b) *SPIN* reasoner: input size computed as number of RDF triples with the original (dark purple) and compressed (light green) input for each data flow.

Figure 4.9: Input size for the *Prolog* (4.9a) and *SPIN* (4.9b) reasoners. It can be deduced that the size of the data flow has a small impact on the general size of the input.



(a) *Prolog* reasoner.



(b) *SPIN* reasoner.

Figure 4.10: Coefficient of Variation (CV) of the observations. Each experiment was run 20 times. In the diagram we see the CV of measures observed. For each data flow, the diagram shows twelve bars. The first six bars refer to the uncompressed rule base, the second six to the compressed rule base. Each group shows the CV for, in order: T, L, S, Q, Pa and M.

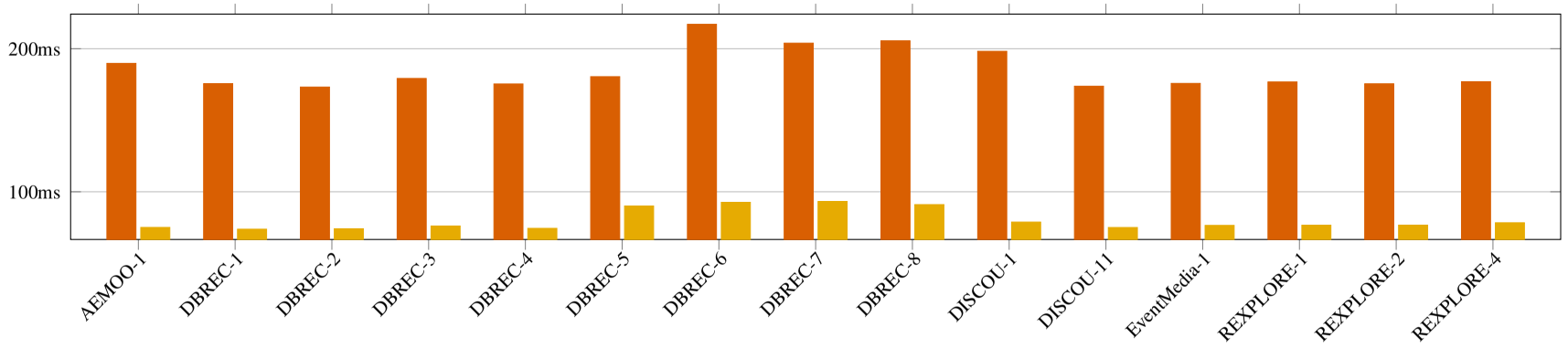
irrelevant. Finally, we consider the computed mean of the observed measures in these experiments to be significant.

Before discussing the results, it is worth reminding the reader that this evaluation is not targeted to compare the two implementations of a PPR reasoner, but to observe the impact of our compression strategy on the *approaches* of the *Prolog* and *SPIN* implementations, assuming that any other implementation is likely to make use of a combination of the two reasoning strategies they respectively implement.

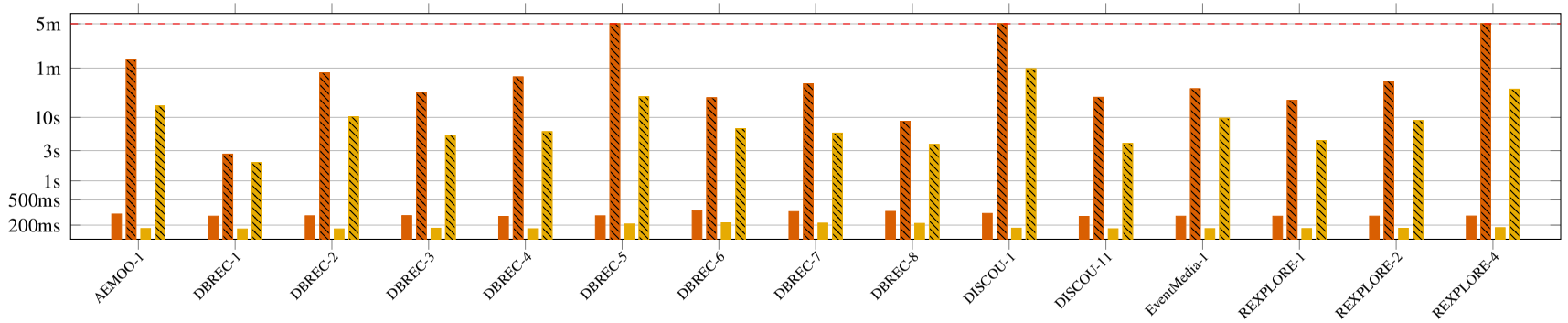
Figures 4.11 and 4.12 illustrate the results of the experiments performed with the *Prolog* and the *SPIN* reasoner, respectively. For each data flow, the bar on the left displays the time with an uncompressed input, and the one on the right the time with a compressed input. We will follow this convention in the other diagrams as well. Figure 4.11c displays a comparison of the total time between an uncompressed and compressed input with the *Prolog* reasoner. In all cases, there has been a significant increase in performance with the compressed rule base: in three cases (DBREC-5, DISCOU-1, REXPLORE-4) the uncompressed version of the experiment could not complete within the five minutes, while the compressed version returned results in less then a minute. The total time of the experiments with the *SPIN* reasoner (Figure 4.12c) is much smaller (fractions of a second), having the maximum total time of approximately 2 seconds (EventMedia-1). However, in this case too, we report an increase in every case in performance for all the data flows, with some cases performing much better than others (DBREC-3, DBREC-4). The total time T of the experiment can be broken up into setup time S (including load time L) and query time Q. This observation is depicted in Figures 4.11a and 4.12a, and in both cases the impact of the rule reduction process is evident. An interesting difference between the two implementations can be seen by comparing Figures 4.11b and 4.12b. The cost of the query time in the *Prolog* reasoner is very large compared to the related setup time S. The *SPIN* reasoner, conversely, showed a larger setup time S with a very low query time Q. The reason is that the latter materialises all the inferences at setup time, before query execution. This accounts for the lack of difference in query time between the uncompressed and compressed version of the experiments with the SPIN reasoner.

We did not observe changes in Pa for the *Prolog* reasoner (Figure 4.11d), while the differences in memory consumption M is significant (Figure 4.11e), demonstrating a performance improvement caused by the compressed input. A decrease in space consumption was also observed using the *SPIN* reasoner (Figures 4.12d and 4.12e), even if smaller, and negative in only 2 cases with regard to memory consumption M (DBREC-1 and DBREC-6).

A summary of the impact of the compression on the different measures is depicted in Figures 4.13 and 4.14. The first bar on the left of both diagrams illustrates the reduction of the size of the *Input*,

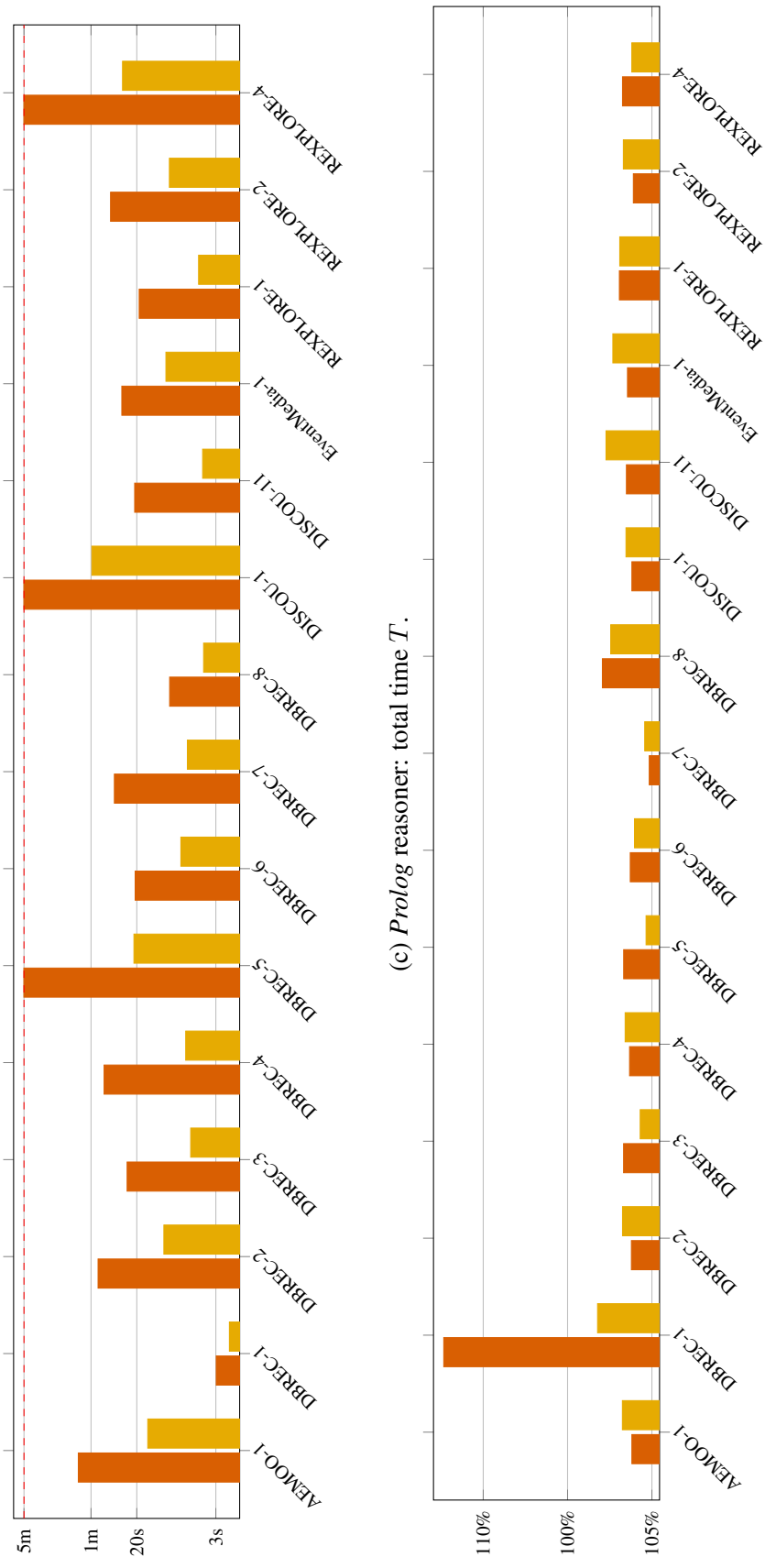


(a) *Prolog* reasoner: load time L.



(b) *Prolog* reasoner: comparison of setup time S and query time Q (bars with pattern).

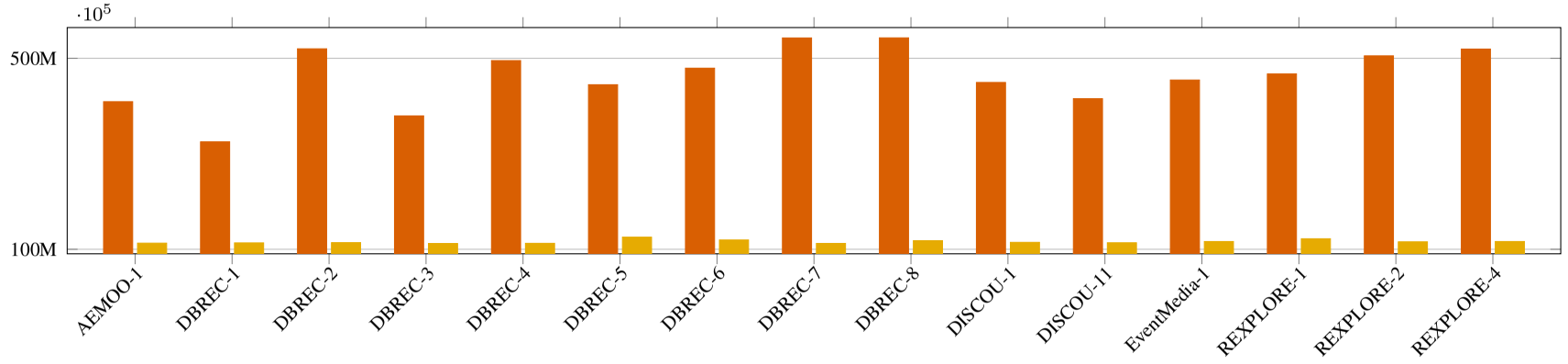
Figure 4.11: *Prolog* reasoner: performance measures (a,b). Each diagram reports on the performance of this reasoner with an uncompressed or compressed rule base with respect to a given measure. The bars on the left (in dark orange) refer to the uncompressed rule base, while the bars on the right (in light yellow) the compressed one.



(c) *Prolog* reasoner: total time T.

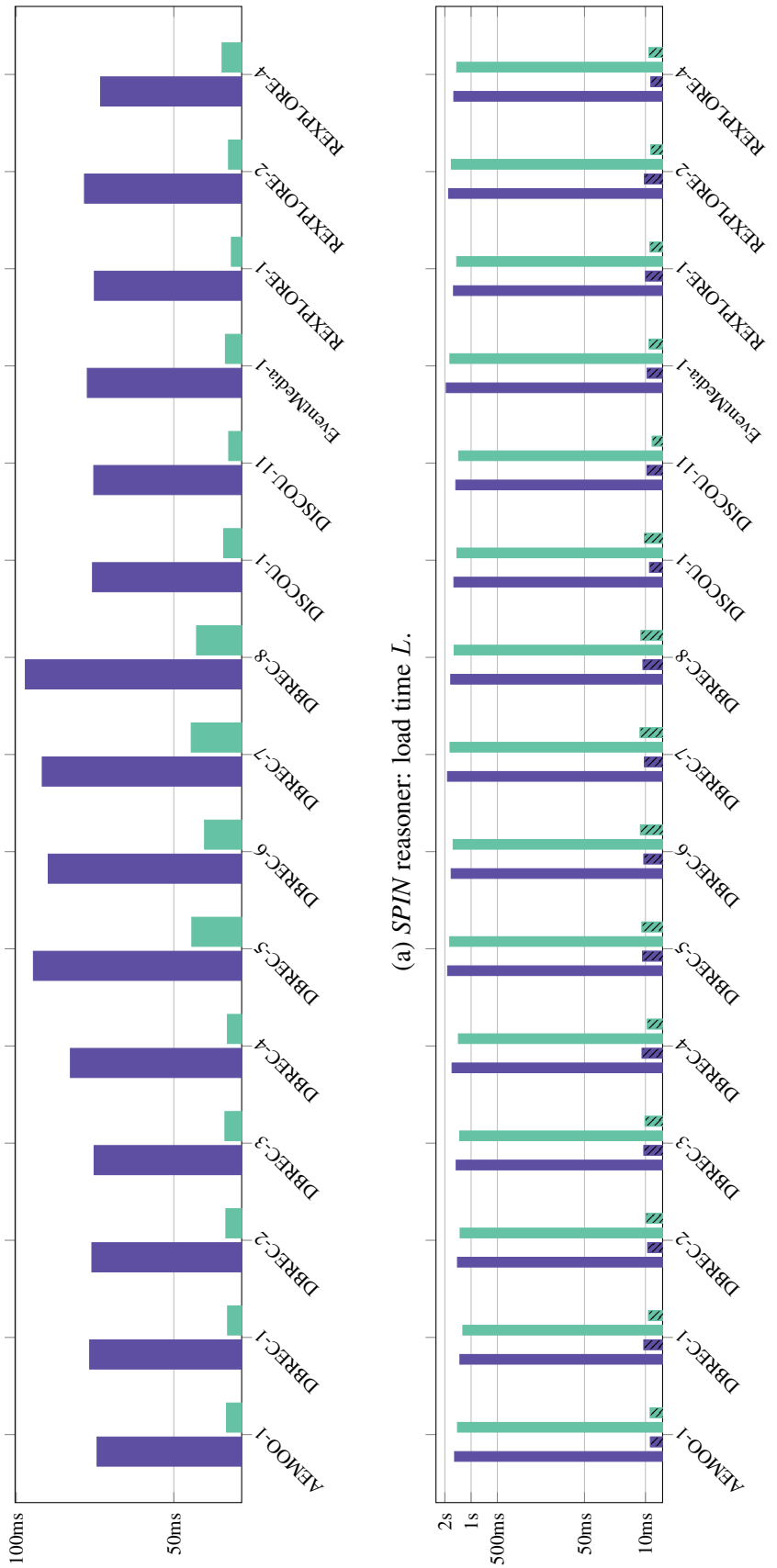
(d) *Prolog* reasoner: average CPU Pa.

Figure 4.11: *Prolog* reasoner: performance measures (c,d). Each diagram reports on the performance of this reasoner with an uncompressed or compressed rule base with respect to a given measure. The bars on the left (in dark orange) refer to the uncompressed rule base, while the bars on the right (in light yellow) the compressed one.



(e) *Prolog* reasoner: max memory consumption M.

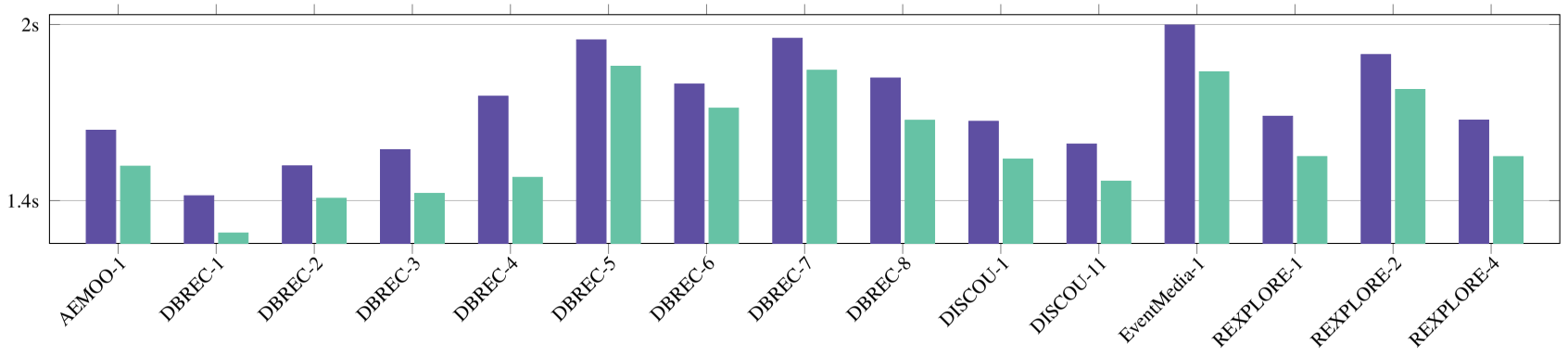
Figure 4.11: *Prolog* reasoner: performance measures (e). Each diagram reports on the performance of this reasoner with an uncompressed or compressed rule base with respect to a given measure. The bars on the left (in dark orange) refer to the uncompressed rule base, while the bars on the right (in light yellow) the compressed one.



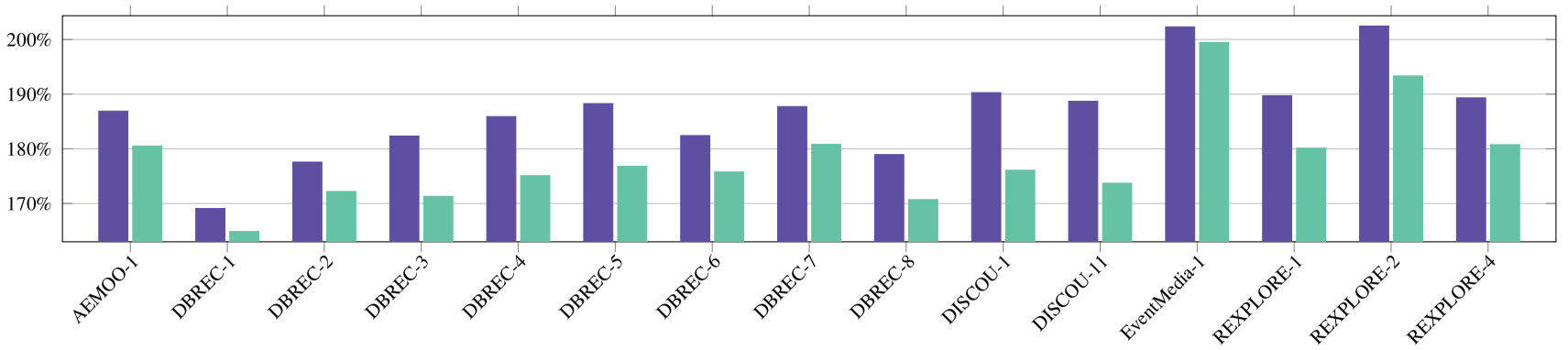
(a) *SPIN* reasoner: load time L.

(b) *SPIN* reasoner: comparison of setup time S and query time Q (bars with pattern).

Figure 4.12: *SPIN* reasoner: performance measures (a,b). Each diagram reports on the performance of this reasoner with an uncompressed or compressed rule base with respect to a given measure. The bars on the left (in dark purple) refer to the uncompressed rule base, while the bars on the right (in light green) the compressed one.

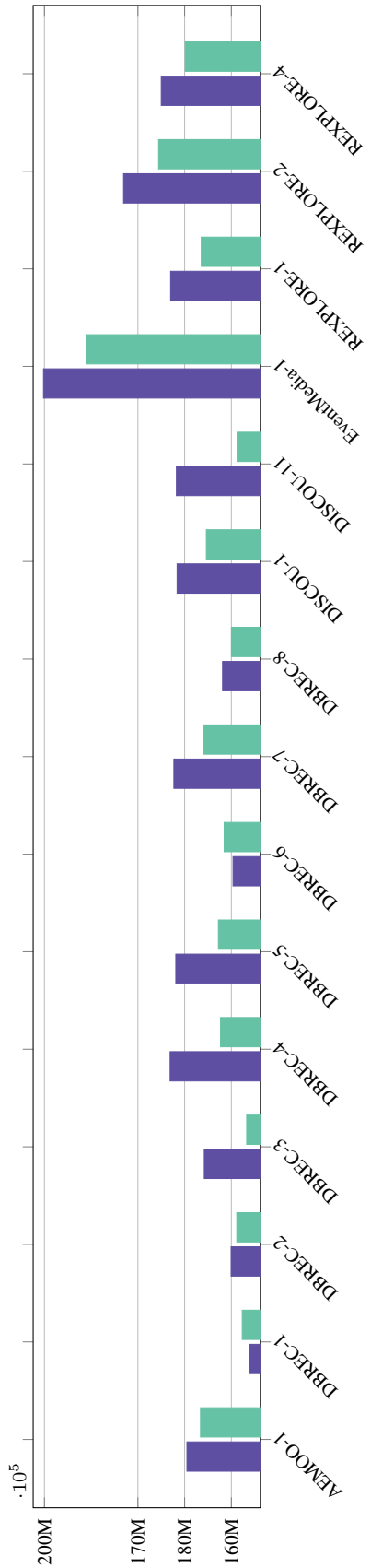


(c) *SPIN* reasoner: total time T.



(d) *SPIN* reasoner: average CPU Pa.

Figure 4.12: *SPIN* reasoner: performance measures (c,d). Each diagram reports on the performance of this reasoner with an uncompressed or compressed rule base with respect to a given measure. The bars on the left (in dark purple) refer to the uncompressed rule base, while the bars on the right (in light green) the compressed one.



(e) *SPIN* reasoner: max memory consumption M.

Figure 4.12: *SPIN* reasoner: performance measures (e). Each diagram reports on the performance of this reasoner with an uncompressed or compressed rule base with respect to a given measure. The bars on the left (in dark purple) refer to the uncompressed rule base, while the bars on the right (in light green) the compressed one.

while the others how much each measure is reduced. A serious improvement has been achieved in the case of the *Prolog* reasoner, implementing a backward chaining algorithm executed at query time. A PPR reasoner could also be implemented to perform inferencing at loading time (materialisation). The experiments with the *SPIN* implementation is therefore used to show that the effect on reasoning performance exists in both cases, even if in different ways depending on the approach to inferencing. The main conclusion from our experiments is therefore that the (A)AAAA methodology presented in the previous Section leads to a compressed PPR knowledge base that is not only more manageable for the knowledge engineers maintaining them, but also improves our ability to apply reasoning for the purpose of policy propagation. In addition, it appears clearly that, when dealing with a compressed PPR knowledge base, an approach based on materialisation of inferences at load time is preferable to one based on computing the inferences at query time.

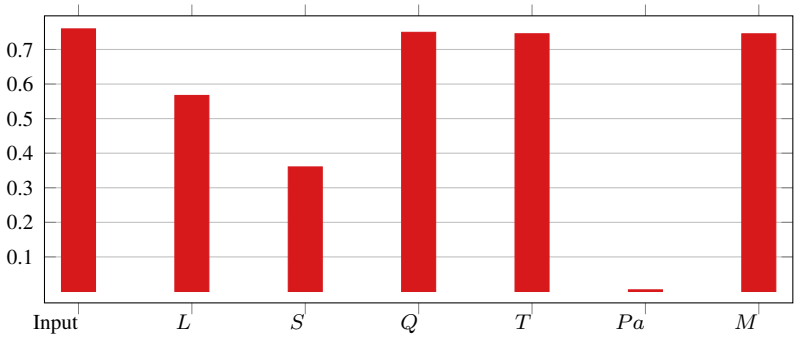


Figure 4.13: *Prolog* reasoner: impact of compression on reasoner performance. The bars show the factor by which each measure has been reduced by applying a compressed input.

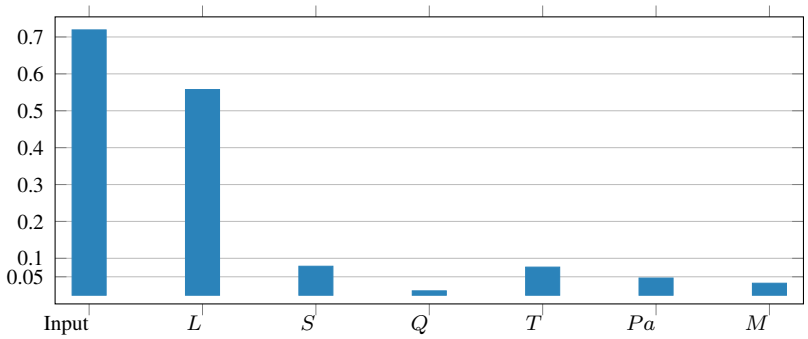


Figure 4.14: *SPIN* reasoner: impact of compression on reasoner performance. The bars show the factor by which each measure has been reduced by applying a compressed input.

4.4 Conclusions

In this Chapter we presented an approach for reasoning on the propagation of policies in a data flow. This method relies on a rule base of Policy Propagation Rules (PPRs). Rules can easily grow in number, depending on the size of the possible policies and of the possible operations performed in a data flow. The (A)AAAA methodology can be used to reduce this size significantly by relying on the inference properties of the Datanode ontology, applied to describe the possible relations between data objects. However, while this activity reduces the size of the input of the reasoner, it requires more inferences to be computed. Therefore, we performed experiments to assess the impact of the compression on reasoning performance.

The approach described in this Chapter clearly relates to principles and methods of knowledge engineering [Studer et al. (1998)]. In [Motta et al. (1990)], knowledge acquisition is considered as an iterative process of model refinement. More recently, problem solving methods have been studied in relation to the task of understanding process executions [Gómez-Pérez and Corcho (2008)]. These contributions form the background of the approach we are following in the present work. The problem of compressing propositional knowledge bases has been extensively studied in the past, focusing on the optimisation of a Horn minimisation process to improve the performance of rule execution [Boros et al. (2013); Hammer and Kogan (1993)]. Differently, we deal with compression as a mean to reduce the number of minimal rules to be managed (each PPR being already an atomic rule), by means of an additional knowledge base (the Datanode ontology). This is also a preliminary step towards studying compression in knowledge management and its impact on reasoning in a more general way. This problem is also different from the one of minimising access control policies (for example, the abstraction by subsumption proposed in [Javanmardi et al. (2006)]), as the abstraction required is on the propagation of the policy, not the policy itself. Reasoning on policy propagation does not require the policies to be validated per se. On the contrary, we claim that validating the policies of a data artefact, which is the result of some manipulation, should consider the policies inherited from the input data, according to the particular actions performed.

We have identified the components for policy propagation: ODRL licences, Datanode processes and PPRs, and demonstrated how with these components reasoning on policy propagation is possible with state of the art computational resources ($R.Q.$ 2). However, at this stage we do not focus on the *user*, and how she/he might interact with these knowledge components. For example, we assumed that a licence is associated with the published data, and that a Datanode description of the process exists. These aspects will be discussed in the next Chapter, among others, where we study how this approach can be integrated in a more general framework for the governance of policies and their propagation in a city Data Hub.

Chapter 5

A Methodology and Toolkit for Data Cataloguers

Thanks to a knowledge engineering approach, in the previous chapters we were able to identify the core components required for policy propagation, answering our primary research question ($R.Q.\ 1$). They are:

(a) a data licence expressed as a set of atomic policies;
(b) a representation of the system in terms of data-to-data relations; and
(c) rules capable of linking data relations and atomic policies, such that a reasoner could conclude what policies propagate by executing them.

We demonstrated that these components are sufficient to reason on policy propagation and that performing such reasoning is practicable ($R.Q.\ 2$) and sustainable with state of the art computational resources ($H.\ 3$). In this chapter, we face an equally important aspect, pertaining to the support we can give to users for developing the needed knowledge components and how they can be integrated into a unified end-to-end solution for solving the task of assessing what policies propagate from input data sources to the output(s) of a data relying activity. To this purpose, we will look at Smart Cities, and how data is at the centre of their more recent developments, and we point at the task of policy propagation assessment as a crucial one in order to decide how derived data can be *exploited* by third parties. Indeed, an end-to-end solution for policy propagation would require those components to be acquired, curated and managed *at scale*. In particular:

(a) data publishers need to associate a licence to their data, by selecting one from a collection of ODRL described licenses (like the ones provided by the RDF Licenses Database);
(b) data managers need to produce a collection of Datanode models representing the computation happening in the data processing infrastructure (like the ones described in Chapter 3);

(c) these components need to be orchestrated among the different actors, and the applicability of the related approach has to be validated in a realistic scenario.

In this chapter, we focus on the end-user and how we can provide support in the acquisition of the knowledge components for policy propagation, and how they can be orchestrated in a unified framework. An end-to-end solution includes:

- a method and a tool to support data publishers in the selection of an appropriate data licences, by relying on a catalogue of licenses described with ODRL;
- a method and a tool to annotate data manipulation processes in order to derive Datanode descriptions demonstrated on scientific workflows as paradigmatic data-intensive process models; and
- a methodology integrating the licences, dataflows and PPRs, clarifying the types of users and their role, and developable with state of the art technologies.

In the next section, we introduce the reference context of a Smart City data infrastructure. In Section 5.2 we focus on the problem of supporting data publishers in choosing the right policies (among a large variety of options), showing how a semantic model can be of use for reducing the effort required for licence selection. Section 5.3 is dedicated to solving the important problem of supporting data managers on producing Datanode process descriptions. Finally, we illustrate a data cataloguing methodology in Section 5.4, demonstrating how to compose all the contributions introduced so far in a unified framework to support policy propagation in a Data Hub.

5.1 Context: the Milton Keynes Data Hub

Smart Cities can be seen as composite systems in which information rapidly flows through multiple devices and data silos [Nam and Pardo (2011); Townsend (2013)]. The amount of data to be managed is rapidly increasing, together with the scenarios, use cases and applications that rely on such shared data. A Smart City **data hub** is an infrastructure that manages a wide range of data sources and methods of delivering them, with the aim of providing users with services that rely on data taken from the sources it manages.

Our work is placed within the context of the MK Data Hub [d'Aquin et al. (2015b)], the data sharing platform of the MK:Smart project[1], which explores the use of data analytics to support Smarter Cities, taking the city of Milton Keynes in the UK as a testbed. The data catalogue of the MK Data Hub contains information about a large number of datasets from many different

[1] see `http://mksmart.org`

sources, including open data from the local council and the UK government, as well as data from private sector organisations (e.g. utility companies) and individuals. The main purpose of the MK Data Hub is to support applications that combine different city data in innovative scenarios. It, therefore, includes data access mechanisms (APIs) that provide an integrated view over the data. However, in order to enable the reuse of such data, not only technical integration mechanisms are required. Indeed, Since the data as a result of the MK Data Hub APIs might be combined from diverse datasets, different parts of the data might have different exploitability conditions or requirements, propagated from the licences and policies associated with the original datasets. A data consumer (and application developer) might, for example, need to filter data for use in a commercial application, discarding any data from sources that explicitly, in the original data licence, specified that such use of the data was prohibited. Similarly, Data consumers might need to check which original sources of the data need to be acknowledged because of an attribution requirement, and even whether the form of exposure or re-distribution they employ is allowed according to the policies attached to each individual piece of data they might obtain from the Data Hub. The issue of exploitability is, therefore, one that directly relates to providing the right level of information regarding the rights and policies that apply to the data being delivered by data hubs. Data sources in the MK Data Hub include sensor data, public data extracted from the Web as well as data provided by public institutions and other organisations, such as the Milton Keynes Council. These data sources, however, come with a set of policies regulating their usage. For example, the "Bletchley and Fenny Stratford" ward is a British electoral division that corresponds to an area in the South of the city. Located within this ward are a number of sensor devices that push data of varied nature to the Data Hub, including Air quality and Soil moisture (see an example in Figure 5.1). The National Museum of Computing is located in Bletchley Park, and it is often a topic of interest in social platforms like Flickr. The Milton Keynes Council provides the MK Data Hub with statistics about population growth, crime, marital status, religion and employment, among others. All these data sources are catalogued, consumed and stored as *datasets* by the Data Hub in order to provide the end-user with services that intensively rely upon these data. One of these services is the *Entity-Centric API* (ECAPI) of the MK Data Hub. The ECAPI offers an entity-based access point to the information offered by the Data Hub, aggregating data from multiple data sources around 'real world entities' such as geographical regions, buildings, bus stops etc [Adamou and d'Aquin (2015)]. The aforementioned ward (see Figure 5.2 for some example data) and museum in Milton Keynes are examples of named entities the ECAPI may be queried for. More generally, any arbitrary geographical area within a fixed radius of given geospatial coordinates (e.g. `51.998,-0.7436` in decimal degrees) could be an entity for an application to try to get

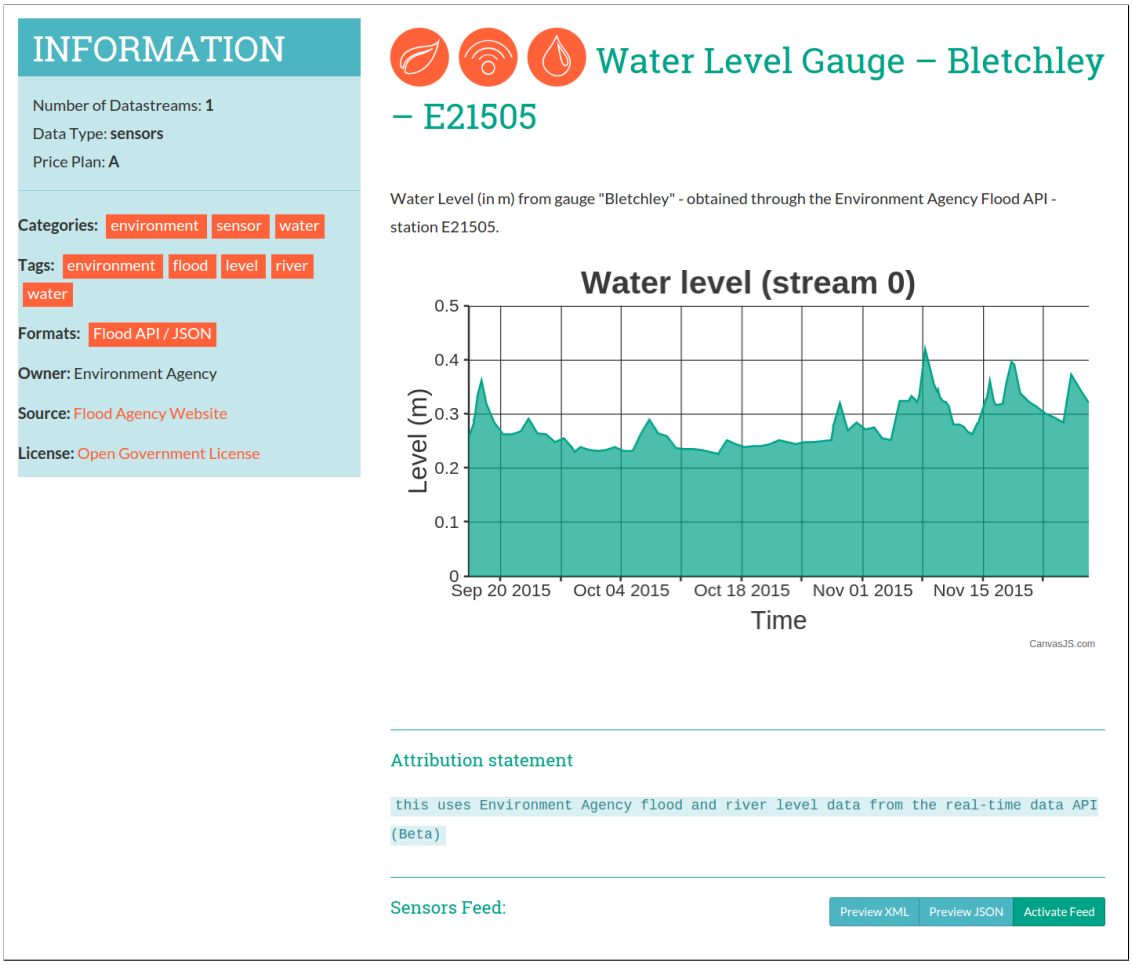


Figure 5.1: MK:Smart Data Hub: the dataset “Water Level Gauge - Bletchley - E21505”.

Figure 5.2: MK:Smart Data Hub: example of a *British ward*.

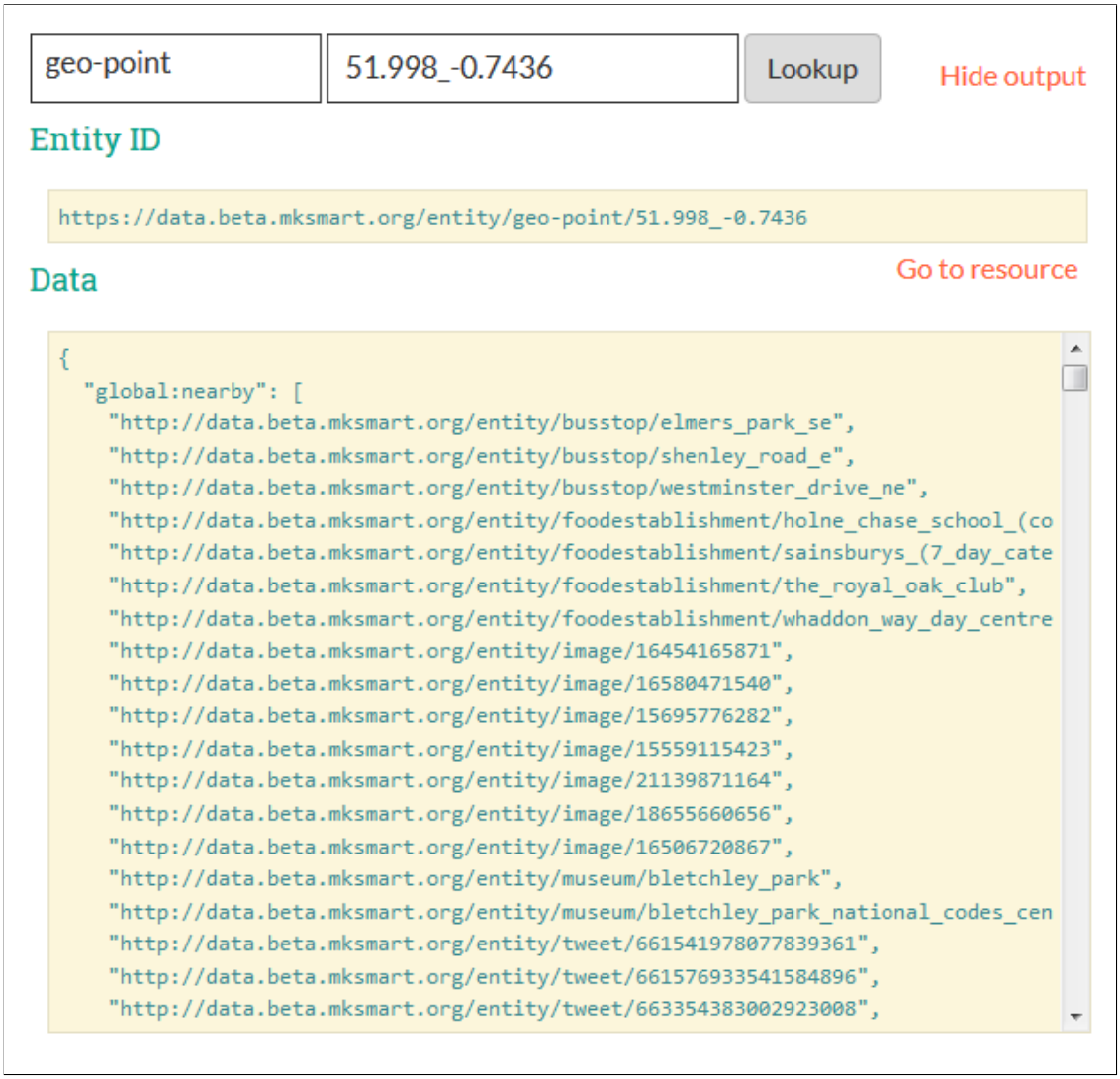


Figure 5.3: MK:Smart Data Hub: example of a geographical point.

information about (see Figure 5.3 for example data). The ECAPI will return a collection of items that are relevant for that location, selected from the appropriate datasets. However, the parts of the returned data have been collected (and processed) from sources that have different usage policies. This makes the exploitation of the data problematic.

Using the MK Data Hub as an example of multi-source Smart City data infrastructure, we consider data **exploitability** to specify the compatibility of the policies attached to the delivered data – obligations, permissions and prohibitions – with the requirements of the user's task. Ultimately, *how to support data publishers, data managers and data consumers in the various tasks associated with assessing the restrictions, permissions and obligation that need to be considered when exploiting the data published by a Smart City Data Hub?*

5.2 Tool support for licence selection

The already introduced RDF Licenses Database [2] is a first notable attempt at developing a knowledge base of licences described in ODRL. However, identifying suitable licences is still not a trivial task for a data publisher. In the current version, ODRL identifies more than fifty possible actions to be used as permissions, prohibitions or obligations, and there are ontologies that extend ODRL adding even more fine grained policies (e.g. LDR[3]). Therefore, not only are there many licences that can be applied, but each might include any subset of the many possible features (permitted, prohibited and required actions), which need to be explored in order to obtain a small selection of comparable licences to choose from.

The question that we aim to answer is: *how can we reduce the effort for licence identification and selection?* We advance the hypothesis that an ontology defining relevant classes of licences, formed on the basis of the *key features* of the instances, should facilitate the selection and identification of a suitable licence. We develop a methodology relying on a *bottom-up* approach to ontology construction based on Formal Concept Analysis (FCA). We developed a tool, Contento, with the purpose of analysing data about licences using FCA, in order to generate a concept lattice. This concept lattice is used as a draft taxonomy of licence classes that, once properly annotated and pruned, can be exported as an OWL ontology and curated with existing ontology editors. We applied this approach to the use case of licence identification, and created a service to support data providers in licence selection by asking a few key questions about their requirements. In this Section we show that, with this service, we can reduce the selection of licences from comparing more than fifty

[2] http://datahub.io/dataset/rdflicense

[3] http://oeg-dev.dia.fi.upm.es/licensius/static/ldr/

possible licence features, to answering on average three to five questions.

Section 5.2.1 describes the process of building the ontology, the Contento tool and the modelling choices that have been made. In Section 5.2.2 we report on the application of the ontology in a service for identification of suitable licences for data providers. Furthermore, we compare existing approaches to solve the problem of licence selection and identification with the proposed solution in Section 5.2.3.

5.2.1 Building the ontology with Contento

Our hypothesis is that an ontology can help on orienting the user in the complex set of existing licences and policies. The RDF Licenses database contains 139 licences expressed in RDF/ODRL. Our idea is therefore to start from the data to create the ontology. The reason for choosing a bottom-up approach to ontology construction is also that the data will include only policies that are relevant.

In order to support the production of the ontology we implemented a bottom-up ontology construction tool called Contento, which relies on FCA. FCA has the capability of classifying collections of objects depending on their *features*. The input of a FCA algorithm is a *formal context* - being a binary matrix having the full set of objects as rows and the full set of attributes as columns. Objects and attributes are analysed and clustered in *closed concepts* by FCA. In FCA, a concept consists of a pair of sets - objects and attributes: the objects being the extent of the concept and the attributes its intent. A subsumption relation can be established between formal concepts in order to define an order on the set of formal concepts in a formal context. As a result, formal concepts are organised in a hierarchy, starting from a top concept (e.g., *Any*), including all objects and an empty set of attributes, towards a bottom concept (e.g., *None*), with an empty set of objects. Moreover, this ordered set forms a mathematical structure: the concept *lattice*.

The objective of the Contento tool is to support the user in the generation and curation of concept lattices from formal contexts (binary matrixes) and to use them as drafts of Semantic Web ontologies.

Contento

Contento[4] has been developed to create, populate and curate FCA formal contexts and associated lattices, also interpreted as taxonomies of concepts. Formal contexts can be created and populated from scratch. Sets of items can be managed with a number of features in the *Collections* section.

[4] `http://bit.ly/contento-tool`

The user can assign the role of objects' set and attributes' set to two collections, thus to generate a formal context. Figure 5.4 presents the formal context browser of *Contento*. Each context is represented as a list of relations between one object and one attribute and a *hold* status: *yes*, *no* or *undefined*. The *undefined* status has been included to indicate that the relation has not been supervised yet. The user can then incrementally populate the formal context by chosing whether each object/attribute association occurs or not. This can be done conveniently thanks to a set of filtering options that can reduce the list to only a subset of the context to be analysed. Data can be filtered in different ways:

– by object name (or all that have a given attribute)
– by attribute name (or all that have a given object)
– by status (holds, does not hold, to be decided)

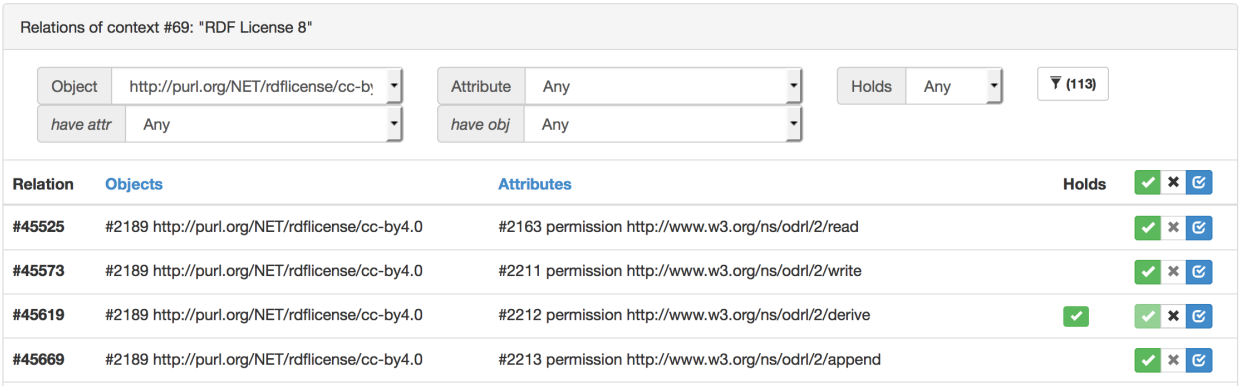


Figure 5.4: Contento: formal context browser and editor. In this example, we have fixed the object in order to review its relations with the attributes.

Therefore, the user can display only the relations that need to be checked (the ones with status *undefined*), focus on the extent of a specific attribute or on the intent of an object. Moreover, she can display the set of relations having for object any that include a specific attribute (or vice versa). Eventually, the user can set all filtered relations to a given state in bulk, if meaningful. With this interface, the binary matrix can be incrementally populated to constitute a proper input for a FCA algorithm.

In many cases, however, a ready made binary matrix can be imported from pre-existing data. In this case the formal context is created directly from that, ready to be used to generate the concept lattice with the procedure provided by Contento.

Contento implements the Chein algorithm [Chein (1969)] to compute concept lattices. The result of the algorithm is stored as a *taxonomy*. A taxonomy can be navigated as an ordered list

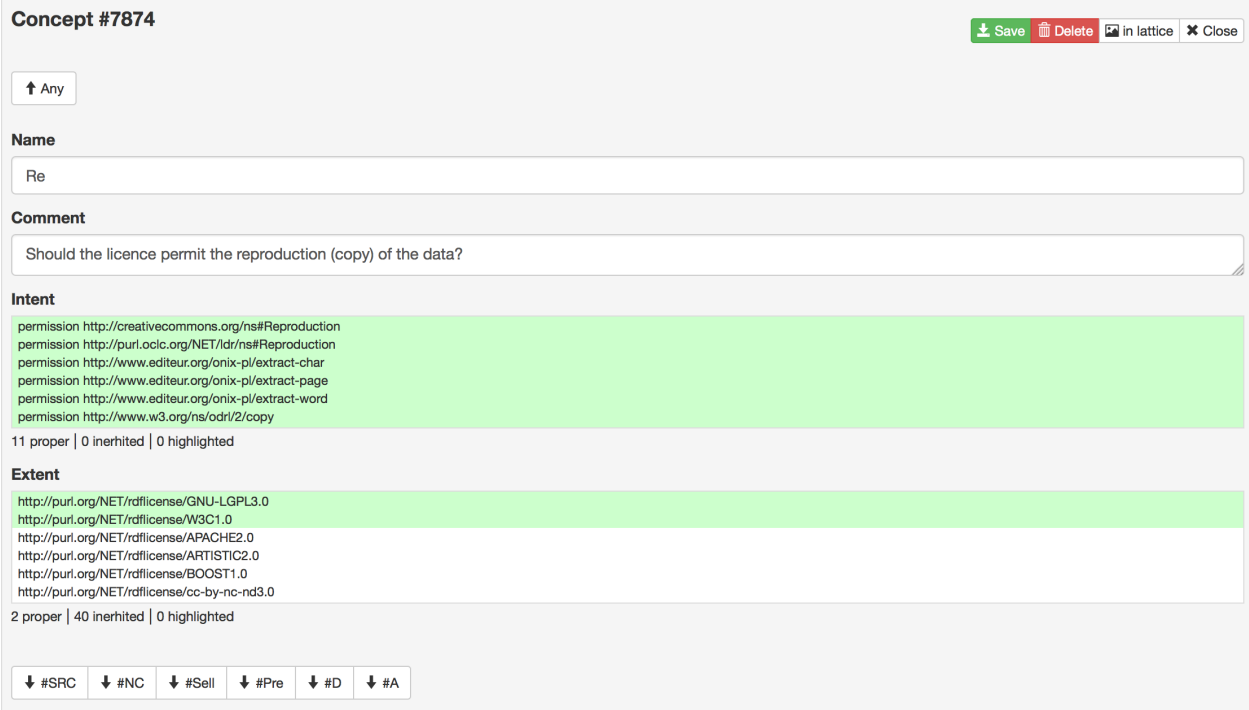


Figure 5.5: Contento: Concept view. Each concept is presented showing the extent, the intent and links to upper and lower concept bounds in the hierarchy. The portion of the intent not included in any of the upper concepts (called *proper*) is highlighted, as well as any object not appearing in lower concepts. Concepts can be annotated and deleted.

of concepts, from the top to the bottom, each of them including the extent, the intent and links to upper and lower concept bounds in the hierarchy (see Figure 5.5). In addition, the tool shows which objects and attributes are *proper* to the concept, i.e. do not exist in any of the upper (for attributes) or lower (for objects) concepts.

Moreover, it can be visualised and explored as a concept lattice (Figure 5.6). The lattice can be navigated by clicking on the nodes. Focusing on a single node, the respective upper and lower branches are highlighted, to facilitate the navigation to the user. Similarly, objects and attributes from the focused node can be selected, thus highlighting all nodes in the hierarchy sharing all of the selected features (in orange in Figure 5.6). Contento supports the user on the curation of the concept hierarchy, to transform it from a concept lattice to a draft ontology taxonomy, through the annotation of each concept with a label and a comment, and the pruning of unwanted concepts. This last operation implies an adjustment of the hierarchy, by building links between lower and upper bounds of the deleted node (only if no other path to the counterpart exists). As a result, relevant concepts can be qualified, and concepts that are not relevant for the task at end can be removed.

Taxonomies can be translated into OWL ontologies. The user can decide how to represent the

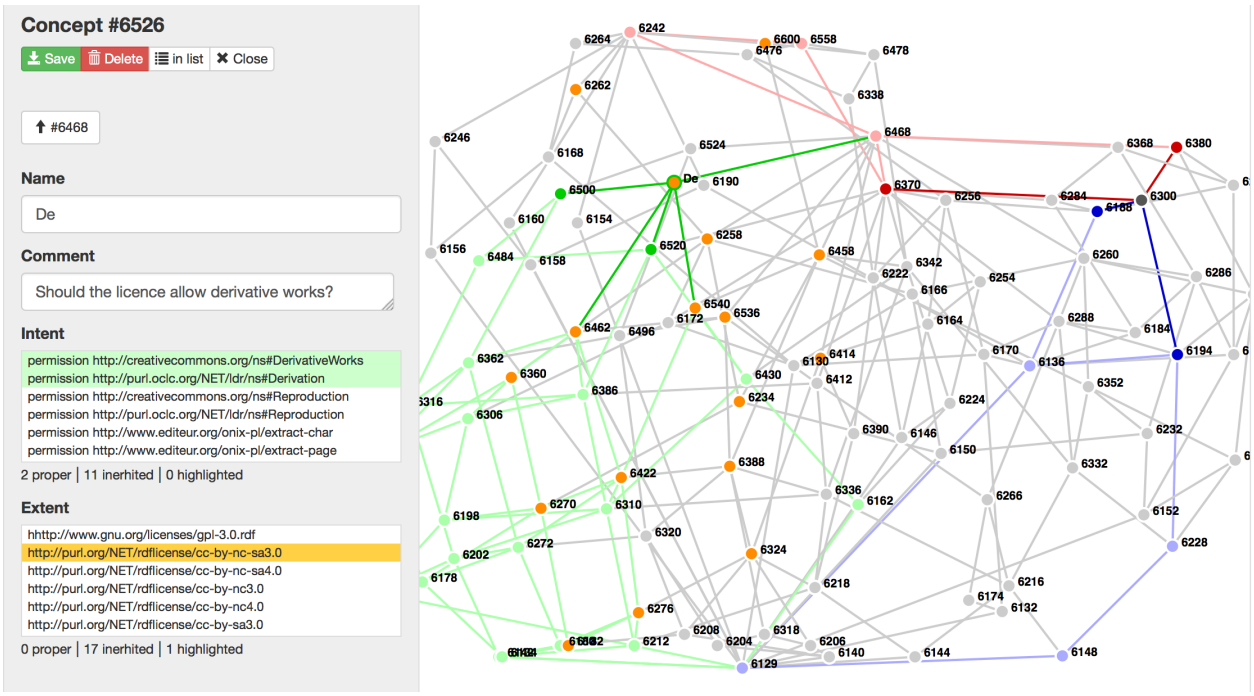


Figure 5.6: Contento: the lattice explorer for annotation and pruning. The branching of the current concept is presented in the lattice in green (on the left side of the picture). The user can still point to other nodes to inspect the branching of other concepts (on the right side of the picture, the lower branch being displayed in blue and the upper in red). By selecting one or more items in the extent or intent of the concept, all the nodes sharing the same are highlighted in orange.

taxonomy in RDF, what terms to use to link concepts, objects and attributes, and whether items need to be represented as URIs or literals. Ultimately, these export configurations can be shared and reused. For example, Contento offers a default profile, using example terms, or a SKOS profile.

The ontology

We used Contento to support the creation of the Licence Picker Ontology (LiPiO)[5], starting from data in the RDF Licenses database. The data has been preprocessed in order to produce a binary matrix to be imported in Contento. The preprocessing included reasoning on SKOS-like relations between ODRL actions[6]. Moreover, we reduced the number of licences from the initial 139 to 45 by removing localised versions (for instance Creative Commons CC-BY-SA 3.0 Portugal). In this case, the licences are the objects of the matrix, while the set of attributes represent the policies, expressed

[5] `http://bit.ly/licence-picker-ontology`

[6] We also introduced some changes in the original descriptions, which will be contributed to evolve the RDF Licence database itself.

as ODRL permissions, prohibitions or duties. Below is an example taken from the input CSV:

```
http://purl.org/NET/rdflicense/cc-by4.0,
 permission http://www.w3.org/ns/odrl/2/copy,
  1
http://purl.org/NET/rdflicense/allrightsreserved,
 prohibition http://www.w3.org/ns/odrl/2/copy,
  1
http://purl.org/NET/rdflicense/MOZILLA2.0,
 duty http://www.w3.org/ns/odrl/2/shareAlike,
  0
```

In the above excerpt, the "CC-BY" licence permits to *copy*, the "All rights reserved" policy prohibits it, and the "Mozilla 2.0" licence does not include a *share-alike* requirement.

The CSV has been imported in the Contento tool that created the formal context automatically. After that, a concept lattice was generated. The lattice included 103 concepts organized in a hierarchy, the top concept representing *All* the licences, while the bottom concept, *None*, includes all the attributes, and no licence. Figure 5.6 shows the lattice as it looked like at this stage of the process. In this phase, the objective is to inspect the concepts and, for each one of them, to perform one of the following actions:

- If the concept is meaningful, name it and annotate it with a relevant question (e.g. "should others be allowed to distribute the work?") in the comment field;
- If the concept is not meaningful or useful, it can be deleted (with the lattice being automatically adjusted).

We judged the meaningfulness of a concept by observing its intent (set of features). If the concept was introducing new features with respect to the upper concepts, then it is kept in the lattice, given a name and annotated with a question. In the case its intent does not include new features (it is a union of the intents of the respective upper concepts), then it is deleted, because the respective licences will necessarily be present in (at least one of) the upper concepts, and no new question need to be asked to identify them. With this process the lattice has been reduced significantly, and proper names and questions have been attached to the remaining concepts (almost 20% of the initial lattice). Figure 5.7 displays the resulting lattice, labels being synthetic names referring to policies/attributes that have been introduced in that point of the hierarchy; i.e. according to the *key features* that define the concept in relation to its parents.

The resulting annotated taxonomy has been exported as OWL ontology as the initial draft of the the Licence Picker Ontology. The draft included a sound hierarchy of concepts. Both concepts (classes) and licences were annotated with the respective set of policies. Because the policies were expressed as plain literal on a generic *has* property (the data being manipulated as

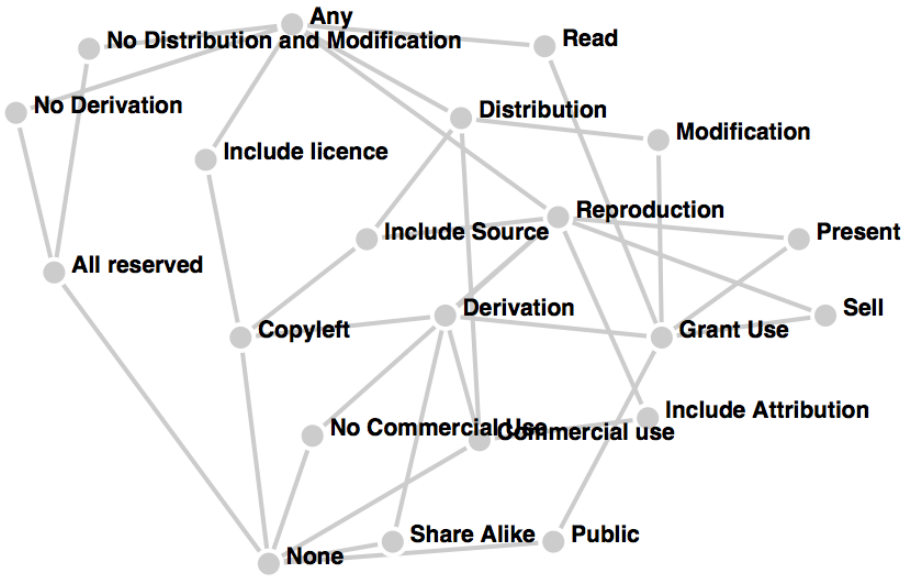


Figure 5.7: Contento: the annotated and pruned concept lattice.

object/attribute pairs by the FCA based tool), a small refactoring permitted to reintroduce the RDF based descriptions with ODRL. The Licence Picker ontology[7] currently contains 21 classes linked to 45 licences with a is-A relation. Each class is associated with a relevant question to be asked that makes explicit the key feature of the included set of licences. The ontology embeds annotations on the classes about the policies included in all the licences of a given concept, and a ODRL based description of permissions, prohibitions and duties of each instance.

5.2.2 Pick the licence

The Licence Picker Ontology has been designed to support data providers in choosing the right policy under which to publish their data. In order to evaluate this ontology we applied it in a service for licence selection. The Licence Picker Webapp is an ontology driven web application[8]. The user is engaged in answering questions regarding her requirements to reach a small set of suitable licences to compare, like in the following guide example. We consider a scenario, inspired from our work on smart cities data hubs [d'Aquin et al. (2014)], in which sensors are installed in a city to detect how busy different areas are at different times, as information to be provided to local retailers, restaurants, etc. This information is collected in a data store and offers access to statistics through a

[7] http://bit.ly/licence-picker-ontology
[8] http://bit.ly/licence-picker-webapp

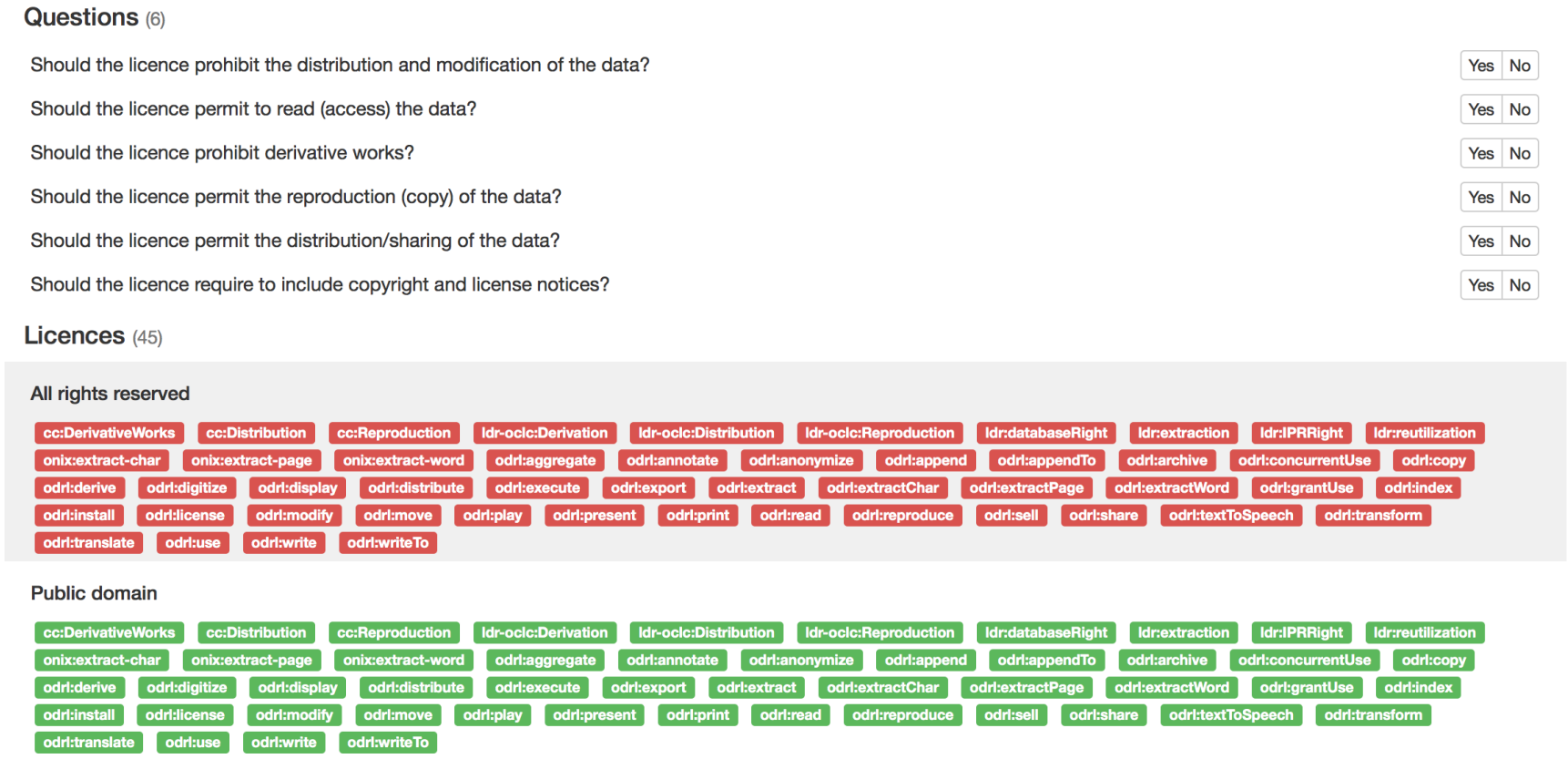


Figure 5.8: Licence Picker Webapp: the user is engaged in answering questions.

number of web-based services. The managers of the data store needs to choose a licence to attach to the data in order to limit their exploitation to the expected uses. They want (a) the data to be accessible and copied for analysis, but (b) to not be modified or redistributed to third parties. In addition, (c) commercial uses should be allowed, but (d) the data consumers should attribute the source of the data to the owner of the data store.

The Licence Picker Webapp welcomes the user with 45 possible licences and a first set of questions, as shown in Figure 5.8. One of them catches the eye of the user: *Should the licence prohibit derivative works?* She promptly answers *Yes*. The set of possible licences is reduced to five, and the system propose a single question: *Should the licence prohibit any kind of use (All rights reserved)?* This time the user answers *No*, because they want the users to use the information to boost the activities in the data store. As a result, the system proposes to pick one of four licences. The user notices that all of them require an attribution statement and prohibit to produce derivative works. Two of them also prohibit the use for commercial purposes, so the user decides to choose the *Creative Commons CC-BY-ND 4.0* licence.

The example above shows an important property of the approach presented in the paper. As the licences are classified by the mean of their features, and the classes organised in a hierarchy, we can notably reduce the number of actions to be taken to obtain a short list of comparable licences. The user had four requirements to fulfil, more then fifty existed in the database, and she could get an easily comparable number of licences with only two steps.

5.2.3 Comparison with related approaches and discussion

Licence recommendation is very common on the web, particularly for software. Services like `http://choosealicense.com/` are usually based on common and well known concerns, and recommend a restricted number of trusted solutions. The Creative Commons Choose service[9] shares with our approach a workflow based on few questions. However, it is an ad-hoc tool which focuses on selecting a Creative Commons licence. In contrast, we are interested in applying a knowledge-based approach, where the way information about licences and requirements is modelled guides the path to the solution. *Licentia* [Cardellino et al. (2014)] is a tool for supporting users in choosing a licence for the web of data. Similarly to our approach, it is based on the RDF licence database. The user selects possible permissions, obligations and duties extracted from the licence descriptions, in order to specify her requirements. The system applies reasoning over the databases of licences, proposing a list of compatible ones to choose from. With this approach the user needs to perform an action for each of its requirements. Our approach restricts the number of questions

[9] `https://creativecommons.org/choose/`

through the inferences implied by the classification of licences in a hierarchy (e.g.: any "share alike" licence allows distribution) and only suggests the ones for which a solution actually exists.

The approach proposed here relies on an ontology of licences as a means for licence selection. Such an ontology has been created following a bottom-up approach. Bottom-up approaches for ontology design have been commonly applied in knowledge engineering [Van Der Vet and Mars (1998)] and we use here one particular method based on Formal Concept Analysis (FCA) [Ganter et al. (2005)]. FCA has been successfully used in the context of recommender systems [du Boucher-Ryan and Bridge (2006); Li and Murata (2010)]. Moreover, FCA has been proposed in the past to support ontology design and other ontology engineering tasks [Cimiano et al. (2004); Obitko et al. (2004)]. In the present work we use FCA as a learning technique to boost the early stage of the ontology design. Licences are an important part of the data publishing process, and choosing the right licence may be challenging. By applying the Licence Picker Ontology (LiPiO), this task is reduced to answering an average of three to five questions (five being the height of the class taxonomy in LiPiO) and assessing the best licence from a small set of choices. We showed how our approach reduces significantly the effort of selecting licences in contrast with approaches based on feature exploration. In addition, a bottom-up approach on ontology building in this scenario opens new interesting challenges. The RDF description of licences is an ongoing work, modelling issues are not entirely solved and we expect the data to evolve in time, including eventually new licences and new types of policies. For example, in our use case the data has been curated in advance in order to obtain an harmonised knowledge base, ready to be bridged to the *Contento* tool. This clearly impacts the ontology construction process and the application relying on it, as different data will lead to different classes and questions. This gives the opportunity to explore methods to automate some of the curation tasks (especially pruning) and to integrate changes in the formal context incrementally, to support the ontology designer in the adaptation of the ontology to the changes performed in the source knowledge base. Such evolutions do not impact the Licence Picker Webapp, because changes in the ontology will be automatically reflected in the tool. We foresee that the description of licences will be extended including other relevant properties - like the type of assets a licence can be applied to. The advantage of the proposed methodology is that it can be applied to any kind of licence feature, not only policies.

The Contento tool was designed to support the task at the centre of the present work. However, the software itself is domain independent, and we plan to apply the same approach to other domains.

5.3 Tool support for data-centric workflow annotations

In the previous chapters a data-centric approach for the representation of data relying systems has been used, the Datanode ontology, with the purpose of simulating the impact of process executions on the data involved, and of performing reasoning on the propagation of data policies. This approach puts the data objects as first class citizens, aiming to represent the possible semantic relations among the data involved. So far, we worked under the assumption that such data-centric descriptions of processes existed within the data infrastructure. In this section we focus on how we can support data managers in producing such descriptions, and we take *Scientific Workflows* as paradigmatic approach to the representation of complex processes, within a Workflow Management System (WMS) as reference component of a Data Hub. However, annotating data intensive workflows is problematic for various reasons: (a) annotation is time consuming and it is of primary importance to support the users in such activity, and (b) workflow descriptions are centred on the processes performed and not on the data, meaning that some form of remodelling of the workflow is required. In this Section we introduce a method based on recommendations to support users in producing data-centric annotations of workflows. Our approach is centred on an incremental rule association mining technique that allows us to compensate the cold start problem due to the lack of a training set of annotated workflows. We discuss the implementation of a tool relying on this approach and how its application on an existing repository of workflows effectively enables the generation of such annotations.

Recently a number of repositories of scientific workflows have been published - Wings[10], My experiments[11], SHIWA[12] are the prominent examples. We selected the My experiments repository as data source for our study. For this reason, we will use the terminology of the SCUFL2 model[13] when discussing how our approach deals with the workflow formalisation.

In the next Section we introduce the approach and Section 5.3.2 how it has been implemented in a tool that allows users to annotate workflows as data-centric descriptions. In Section 5.3.3 we present the results of an experiment performed with real users where we measured how this method impacts the sustainability of the task. Finally, we discuss some open challenges and derive some conclusions in the final Section 5.3.4.

[10]Wings: `\T1\textemdashhttp://www.wings-workflows.org/`.
[11]My experiments: `http://www.myexperiment.org/`.
[12]SHIWA: `http://www.shiwa-workflow.eu/wiki/-/wiki/Main/SHIWA+Repository`
[13]SCUFL2: `https://taverna.incubator.apache.org/documentation/scufl2/`.

5.3.1 Approach

Here we introduce an incremental learning method for the annotation of workflows. Figure 5.9 provides an overview of the approach by listing the elements and their dependency, organised in four phases.

Phase 1. The starting point is an encoded artefact representing the workflow structure and its metadata (like the ones available through My experiments). The workflow code is first translated into a data centric graph, where nodes are data objects manipulated by processors and arcs the relations among them. The result of this transformation is a directed graph with anonymous arcs (named IO port pairs in the Figure), being these arcs the items to be annotated by the user.

Phase 2. Each IO port pair is then associated with a set of features automatically extracted from the workflow metadata.

Phase 3. Extracted features constitute the input of the recommendation engine, designed using the Formal Concept Analysis (FCA) framework. This method is an incremental association rules mining technique that exploits incoming annotations to incrementally produce better recommendations.

Phase 4. Features of the IO port pair, alongside the workflow documentation and the recommendations, are the input for the user who is requested to select a set of annotations from a fixed vocabulary (the Datanode ontology).

First, we focus on the refactoring of a workflow to a data graph, then we introduce the features extraction method and the recommendation engine. We leave the last phase to the next section, where we describe how this approach has been implemented.

Workflows as data-centric graphs

Workflows are built on the concept of *processor* as unit of operation[14]. A *processor* is made of one or more input and output *ports*, and a specification of the operation to be performed. Processors are then linked to each other through a set of data links connecting an output port to the input of

[14]In this paper we use the terminology of the SCUFL2 specification. However, the basic structure is a common one. In the W3C PROV-O model this concept maps to the class *Activity*, in PWO with *Step*, and in OPMW to *WorkflowExecutionProcess*, just to mention a few examples.

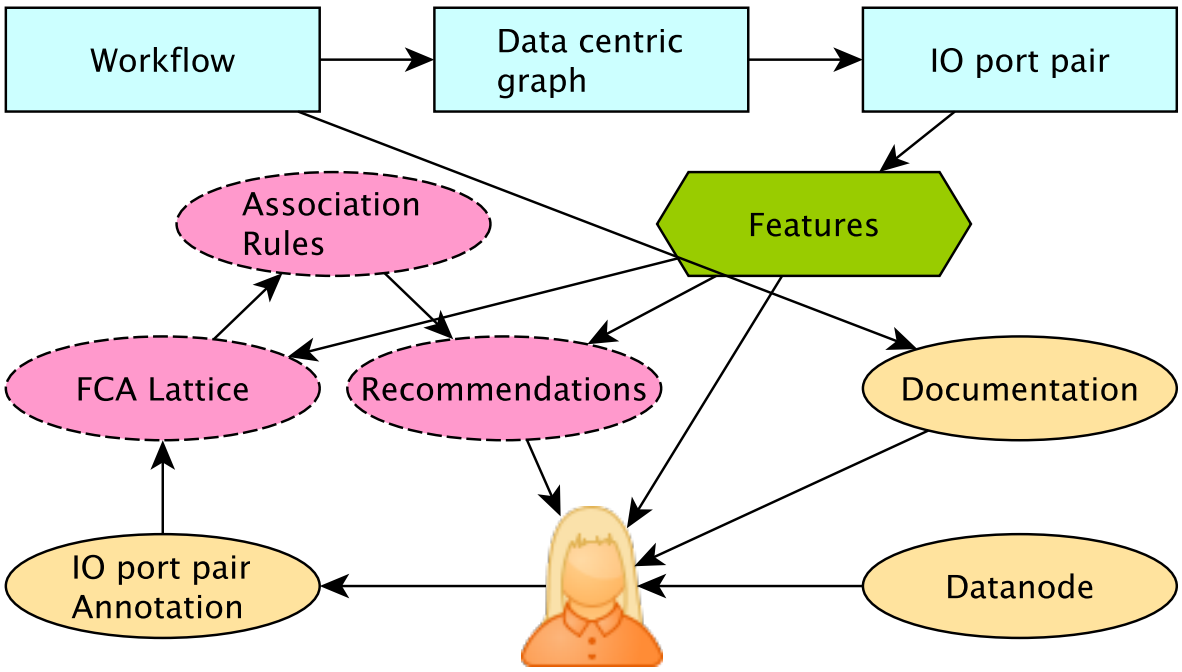


Figure 5.9: Description of the approach and dependencies. Elements of phase 1 are represented in blue rectangles on top. Phase 2 includes the features generation (the only stretched exagon). Elements of Phase 3 are depicted as pink ovals with dashed borders and phase 4 ones as light yellow ovals.

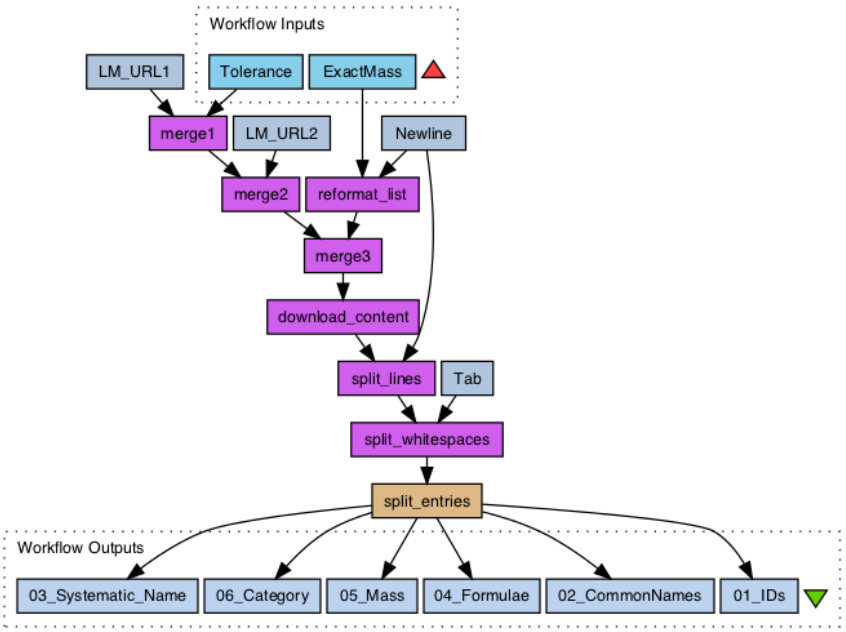


Figure 5.10: A workflow from the My Experiment repository: "LipidMaps Query".

another processor resulting in a composite tree-like structure. Figure 5.10 shows an example of a workflow taken from the "My Experiment" repository[15].

The objective of our work is to describe what happens inside the processors by expressing the relation between *input* and *output*. For example, the processor depicted in Figure 5.11 has two input ports (1 and 2) and one output (3). For this processor, we generate two links connecting the input data objects to the output one, through two anonymous arcs: $1 \rightarrow 3$ and $2 \rightarrow 3$. We name these arcs "IO port pairs" (input-output port pairs), and these are the items we want to be annotated. In this example, the IO port pair $1 \rightarrow 3$ could be annotated with the Datanode relation $refactoredInto$, while the IO port pair $2 \rightarrow 3$ would not be annotated as only referring to a configuration parameter of the processor and not to an actual data input. For the present work we translated 1234 Workflows from the My Experiments repository, resulting in 30612 IO port pairs (altough we will use a subset of them in the user evaluation).

Extracting features from workflow descriptions

As described previously, the workflow description is translated into a graph of IO port pairs connected by unlabelled links. In order to characterise the IO port pair we exploit the metadata

[15]"LipidMaps Query" workflow from My experiment: `http://www.myexperiment.org/workflows/1052.html`.

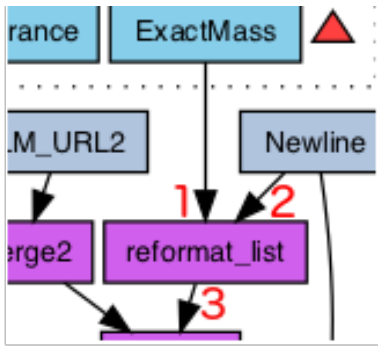


Figure 5.11: Example of workflow processor. This processor has three ports: two input ports (1 and 2) and one output port (3). We can translate this model into a graph connecting the data objects of the inputs to the one of the output.

associated with the components of the workflow involved: the input and output port and the processor that includes them. For each of these elements we extract the related metadata as key/value pairs, which we use as core features of the IO port pair. Applying this approach to the My Experiments corpus we obtained 26900 features. Table 5.1 shows an example of features extracted for the IO port pairs described in Figure 5.11.

However, the objective of these feature sets is to support the clustering of the annotated IO port pair through finding similarities with IO port pairs to be annotated. At this stage of the study we performed a preliminary evaluation of the distribution of the features extracted. We discovered that very few of them were shared between a significant number of port pairs (see Figure 5.12). In order to increase the number of shared features we generated a set of *derived features* by extracting bags of words from lexical feature values and by performing Named Entity Recognition on the features that constituted textual annotations (labels and comments), when present. Moreover, from the extracted entities we also added the related DBPedia categories and types as additional features. As example, Table 5.2 shows a sample of the bag of words and entities extracted from the features listed in the previous Table 5.1.

The generation of derived features increased the number of total features significantly (up to 59217), while making the distribution of features less sparse, as reported in Figure 5.13.

Retrieval of association rules and generation of recommendations

Generating recommendations usually requires an annotated corpus to be avaialable as training set. While repositories of workflows (especially scientific workflows) exist, they are not annotated with data-to-data relations. In order to overcome this problem we opted for an incremental approach, where the recommendations are produced according to the available annotated items *on demand*.

Table 5.1: Sample of the features extracted for the IO port pair $1 \rightarrow 3$ in the example of Figure 5.11.

Type	**Value**
From/FromPortName	string
To/ToPortName	split
Activity/ActivityConfField	script
Activity/ActivityType	`http://ns.taverna.org.uk/2010/activity/beanshell`
Activity/ActivityName	reformat_list
Activity/ConfField/derivedFrom	`http://ns.taverna.org.uk/2010/activity/localworker/org.embl.ebi.escience.scuflworkers.java.SplitByRegex`
Activity/ConfField/script	List split = new ArrayList();if (!string.equals("")) { String regexString = ","; if (regex != void) ...
Processor/ProcessorType	Processor
Processor/ProcessorName	reformat_list

Table 5.2: Example of derived features (bag of words and DBPedia entities) generated for the IO port pair $1 \rightarrow 3$.

Type	**Value**
From/FromPortName-word	string
To/ToPortName-word	split
From/FromLinkedPortDescription-word	single
From/FromLinkedPortDescription-word	possibilities
From/FromLinkedPortDescription-word	orb
From/FromLinkedPortDescription-word	mass
FromToPorts/DbPediaType	wgs84:SpatialThing
FromToPorts/DbPediaType	resource:Text_file
FromToPorts/DbPediaType	resource:Mass
FromToPorts/DbPediaType	Category:State_functions
FromToPorts/DbPediaType	Category:Physical_quantities
FromToPorts/DbPediaType	Category:Mathematical_notation

The rules needed are of the following form:

$$(f^1, f^2, ..., f^n) \rightarrow (a^1, a^2, ..., a^n)$$

where $f1, .., fn$ are the features of the IO port pairs and $a1, .., an$ are the data-to-data relations used to annotate them. Our approach relies on extracting association rules from a concept lattice built through FCA incrementally. Such a lattice is built on a formal context of items and attributes. In FCA terms, the items are the IO port pairs and the attributes their features as well as the chosen annotations. Each node of the FCA lattice is a closed concept, mapping a set of items all having

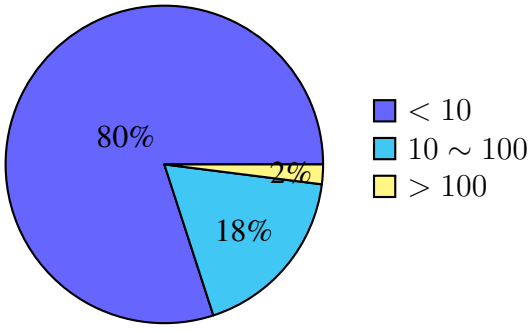


Figure 5.12: Distribution of features extracted from the workflow descriptions.

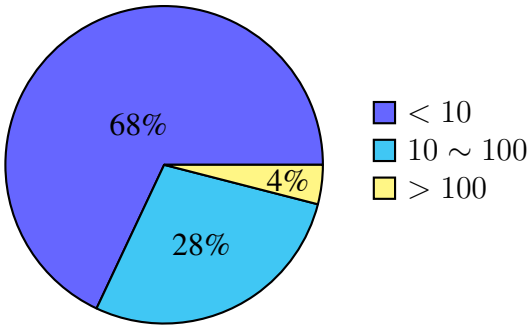


Figure 5.13: Distribution of features (including derived features).

a given set of attributes. A FCA concept would then be a collection of IO port pairs all having a given set of features and/or annotations. In a FCA lattice, concepts are ordered from the top concept (supremum), including all items and (usually) no shared features, to the bottom concept (infimum), including all the available features and a (possibly) empty set of items. The lattice is built incrementally using the Godin algorithm [Godin et al. (1995)]. The algorithm (re)constructs the lattice integrating at each iteration a new item - the IO port pair, with its set of attributes (the features and annotations altogether). Association rules are extracted from the FCA lattice, where the key point is the co-occurrence of features f and annotations a in the various FCA concepts.

The following Listing 5.1 gives a sample of an association rule we want to mine from the lattice:

Listing 5.1: Example of association rule mined from the FCA lattice.

```
(ProcessorName-word: base,
 FromPortName: base64,
 ActivityName-word: decode,
 ActivityType: http://ns.taverna.org.uk/2010/activity/
    beanshell,
 ProcessorName-word: array,
 FromPortName-word: base64,
 ToPortName: bytes,
 ActivityName-word: 64,
 ActivityConfField: mavenDependency,
 ActivityName-word: array,
 ActivityConfField: derivedFrom,
 ProcessorName-word: decode,
```

```
ActivityName-word: byte)
  → (dn:hasDerivation, dn:refactoredInto)
```

Several approaches have been studied to generate and rank association rules from a FCA lattice. A common problem in this scenario is the number of rules that can be extracted, and how to reduce them effectively [Poelmans et al. (2010)]. Indeed, the number of rules can increase significantly with the number of concepts of the FCA lattice. Generating all of them is time consuming as the lattice becomes larger. Precomputing the rules is not a valid solution, as the lattice will change for any new item inserted. In this scenario, we are forced to compute the rules *live* for each new item to be annotated.

The above considerations motivate a set of new requirements for implementing a rule mining algorithm that is effective in this scenario:

1. generate only rules that have annotations in the body
2. generate only rules that are applicable to the candidate item to be annotated
3. only use one rule for each recommendation (head of the rule), to avoid redundancies
4. rank the rules to show the most relevant first

In order to satisfy the requirements above we propose an algorithm to mine association rules *on-demand*, by considering two sets of attributes as constraints for the head and body of the rules.

The algorithm we propose has three inputs: (1) a FCA Lattice; (2) the set of attributes of the item for which we need recommendations (the set of attributes that needs to be in the body of the rules); and (3) the set of attributes we want to be part of the recommendations (the set of attributes that can be in the rule head). Listing 5.2 illustrates the algorithm for extracting rules *on-demand*. The input is a lattice L, a set of attributes as possible recommendations (target rule head: H) and a set of attributes for which we need recommendations (target rule body: B). The algorithm assumes the two sets to be disjoint. The algorithm traverses the lattice starting from the bottom, adding the *infimum* to a FIFO queue - lines 3-5. For each concept in the queue, it first assesses whether its attributes contain items from both the target head and body. If it doesn't, the concept (and related paths in the lattice) can be skipped - lines 7-11. Otherwise, the parent concepts are added to the queue, and the concept considered for rule extraction - line 13. The non empty intersections of attributes with the target head and body form a candidate rule $b \rightarrow h$.

Listing 5.2: Algorithm to mine association rules from a lattice *on demand*:

```
1 // L: the lattice; H: attributes in the rule head; B: attributes in the
     rule body
```

```
2  mineRules(L,H,B):
3    C ← [] // an empty FIFO list of concepts
4    R ← [] // an empty set of Rules (indexed by their head).
5    add(inf(L), C) // add the infimum of L to C
6    while !empty(C):
7       c ← first(C) // remove one concept from the top of the queue
8       h=retain(attributes(c),H) // attributes in c in the head of rule
9       if empty(h): continue // move to another concept
10      b ← retain(attributes(c),B) // attributes in c allowed in the body of
            the rule
11      if empty(b): continue // move to another concept
12      // Add the concept parents to the queue.
13      addAll(parents(L,c),C)
14      // Examine b → h measures (s: support, k: confidence, r: relevance)
15      // support (s): items satisfying the rule divided by all items
16      s ← count(objects(c)) / count(objects(supremum(L)))
17      if s = 0: continue // A supremum rule includes this one
18      // confidence (k): support divided by the items only satisfying b
19      I ← [] // items only satisfying the body
20      for p in parents(c):
21       if (attributes(p) ∩ h) = ∅:
22         if attributes(p) = b: add(objects(p), I)
23       end
24      end
25      if count(I) = 0: k ← 1
26      else:
27        k ← count(objects(c)) / count(I)
28      end
29      // relevance (r): intersection of B with b, divided by B
30      r ← count(B ∩ b) / count(B)
31      // check this rule is the best so far with this head
32      if hasRuleWithHead(R,h):
33         rule ← getRuleWithHead(R,h)
34         if relevance(rule) > r: continue
35         if relevance(rule) = r:
36             if confidence(rule) > k: continue
37             if confidence(rule) = k:
38                 if support(rule) >= s: continue
39             end
40         end
41      end
42      rule ← (h,b,s,k,r) // the new rule, or the best so far for head
43      add(rule, R)
44   end
45 return R
```

The association rule derived is scored by support (s), confidence (k) and a third measure inspired from information retrieval and called *relevance* (r) - lines 15-30. The definitions of these measures, considering a rule $b \rightarrow h$, is as follows:

– Support s ($b \rightarrow h$): the ratio of items satisfying $b \cup h$ to all the items in the lattice - line 16;

– Confidence k ($b \rightarrow h$): the ratio of items satisfying $b \cup h$ to the items satisfying b - lines 19-28;
– Relevance r ($b \rightarrow h$): the degree of overlap between the body of the rule b and the set of features of the candidate item B. It is calculated as the size of the body divided by the size of the intersection between the body of the rule and the features of the candidate item - line 30.

Only the rule with best score for a given head is kept in the list of rules - lines 31-43. Our ranking algorithm will privilege relevance over confidence and support, in order to boost the rules (recommendations) that are more likely to be relevant for the candidate item.

Since this is an iterative process, at the very beginning there will be no recommendation. New annotations will feed the reference corpus (the FCA lattice) and the system will start to generate association rules. Our hypothesis is that the quality of the rules and therefore their usefulness in supporting annotations, increase with the size of the annotated items (this will be part of the evaluation in Section 5.3.3).

5.3.2 Implementation of the approach

The approach described in the previous Section has been implemented in the Dinowolf (Datanode in workflows) tool[16] based on the SCUFL2 worfklow specification[17] and the taxonomy of data-to-data relations represented by the Datanode ontology. While Dinowolf has been implemented leveraging the Apache Taverna[18] library, it can work with any input following the SCUFL2 specification. When a workflow is loaded, the system performs a preliminary operation to extract the IO port pairs and to precompute the related set of features following the methods described in Sections 5.3.1 and 5.3.1. In order to expand the feature set with derived features - bag of words and entities from DBPedia - the system relies on Apache Lucene[19] for sentence tokenization (considering english stopwords), DBPedia Spotlight[20] for named entity recognition, and the DBPedia[21] SPARQL endpoint for feature expansion with categories and entity types. The tool includes three views: 1) a Workflows view, listing the workflows to be annotated; 2) a Workflow details view, including basic information and a link to the external documentation at My Experiments; and a 3) Annotation view, focused on providing details of the features of the IO port pair to annotate. The task presented to the users is the following:

1. Choose an item from the list of available workflows;

[16]Dinowolf: `http://github.com/enridaga/dinowolf`.
[17]SCUFL2 Specification: `https://taverna.incubator.apache.org/documentation/scufl2/`.
[18]Apache Taverna: `https://taverna.incubator.apache.org/`.
[19]Apache Lucene: `https://lucene.apache.org/core/`
[20]DBPedia Spotlight: `http://spotlight.dbpedia.org/`.
[21]DBPedia: `http://dbpedia.org/`.

2. Select an IO port pair to access the Annotation view;
3. The annotation view shows the features associated with the selected IO port pair alongside a list of data node relationships exploiting a set of rules extracted from the FCA lattice as recommendations, and the full Datanode hierarchy as last option;
4. The user can select one or more relations by picking from the recommended ones or by exploring the full hierarchy. Recommended relations, ranked following the approach described in Section 5.3.1, are offered with the possibility to expand the related branch and select one of the possible subrelations as well;
5. Alternatively, the user can skip the item, if the IO port pair does not include two data objects (it is the case of a configuration parameter set as input for the processor);
6. Finally, the user can postpone the task if she feels unsure about what to choose and wants first to explore other IO port pairs of the same workflow;
7. The user iteratively annotate all the port pairs of a workflow. At each iteration, the system makes use of the previous annotations to recommend the possible relations for the next selected IO port pair.

This system has been used to perform the user-based experiments that constitute the source of our evaluation.

5.3.3 Experimental evaluation

Our main hypothesis is that the approach presented can boost the task of annotating workflows as data-to-data annotated graphs. In particular, we want to demonstrate that the quality of the recommendations improves while the annotated cases grow in number. In order to evaluate our approach we performed a user-based evaluation. We loaded twenty workflows from "My Experiments"[22] in Dinowolf and asked six users to annotate the resulting 260 IO port pairs. The users, all members of one research team, have skills that we consider similar to the ones of a data manager, for example in the context of a large data processing infrastructure like the one of [Daga et al. (2016a)]. In this experiment, users were asked to annotate each one of the IO port pairs with a semantic relation from a fixed vocabulary (the Datanode ontology), by exploiting the workflow documentation, the associated feature set and the recommendations provided. The workflows were selected randomly and were the same for all the participants, who were requested also (a) to follow the exact order proposed by the tool, (b) to complete all portpairs of a workflow before moving to the next; (c) to only perform an action when confident of the decision, otherwise to postpone the choice (using

[22]My Experiments: `http://www.myexperiments.org`.

the "Later" action); (d) to select the most specific relation available - for example, to privilege *processedInto* over *hasDerivation*, when possible. Each user worked on an independent instance of the tool (and hence lattice) and performed the annotations without interacting with other participants. During the experiment the system monitored a set of measures:

- the time required to annotate an IO port pair;
- how many annotations were selected from recommendations;
- the average rank of the recommendations selected, calculated as a percentage of the overall size of the recommendation list; and
- the average of the relevance score of the recommendations selected.

Figures 5.14-5.17 illustrate the results of our experiments with respect of the above measures. In all diagrams, the horizontal axis represents the actions performed in chronological order, placing on the left the initial phase of the experiment going towards the right until all 260 IO port pairs were annotated. The diagrams ignore the actions marked as "Later", resulting on few jumps in users' lines, as we represented in order all actions including at least one annotation from at least a single user. Figure 5.14 shows the evolution of the time spent by each user on a given annotation page of the tool before a decision was made. The diagram represents the time (vertical axis) in logarithmic scale, showing how, as more annotations are made and therefore more recommendations are generated, the effort (time) required to perform a decision is reduced. Figure 5.15 illustrates the progress of the ratio of annotations selected from recommendations. This includes cases where a subrelation of a recommended relation has been selected by the user. While it shows how recommendations have an impact from the very beginning of the activity, it confirms our hypothesis that the cold-start problem is tackled through our incremental approach. Figure 5.16 depicts the average rank of selected recommendations. The vertical axis represents the score placing at the top the first position. This confirms our hypothesis that the quality of recommendations increases, stabilizing within the upper region after a critical mass of annotated items is produced, reflecting the same behavior observed in Figure 5.15. Finally, we illustrate in Figure 5.17 how the average relevance score of picked recommendations changes in time. The relevance score, computed as the portion of features matching a given recommendation that overlaps with the features of the item to be annotated, increases partly because the rules become more abstract (contain less features), partly reflecting the behavior of the ranking algorithm and matching the result of Figure 5.16.

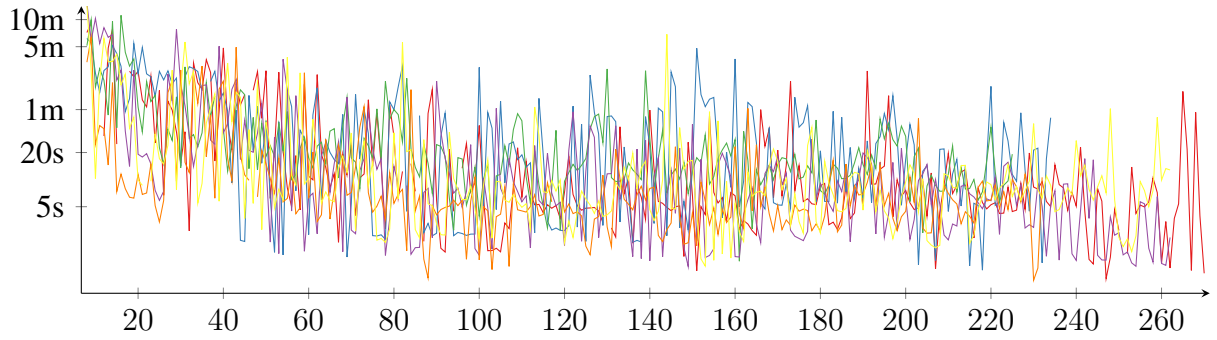


Figure 5.14: Evolution of the time spent by each user on a given annotation page of the tool before a decision was made.

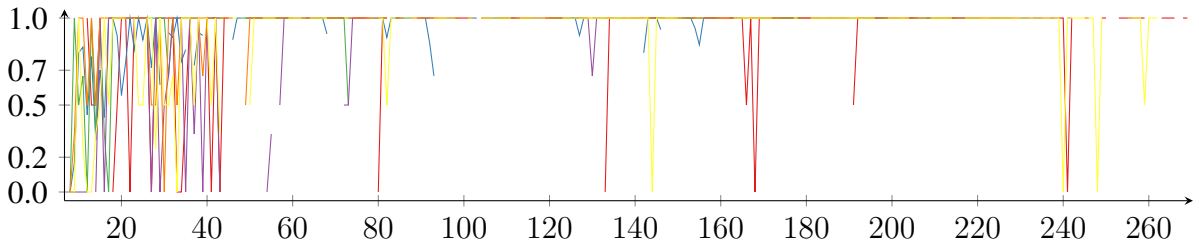


Figure 5.15: Progress of the ratio of annotations selected from recommendations.

5.3.4 Discussion

We proposed a novel approach to support the semantic annotation of workflows with data centric relations. We showed through applying this approach on a set of workflows from the My Experiments repository that it can effectively reduce the effort required to achieve this task for data managers and workflow publishers.

There are several approaches to recommendation using clustering techniques (Support Vector Machines (SVM), Latent Semantic Aanalysis (LSA), to name a few). Formal Concept Analysis (FCA) [Wille (2005)] found a large variety of applications [Poelmans et al. (2013b)], and the literature reports several approaches to incremental lattice construction [Kuznetsov and Obiedkov (2002)], including the Godin [Godin et al. (1995)] algorithm, used in the present work. FCA found application in knowledge discovery as a valuable approach to association rule mining (ARM) [Poelmans et al. (2013b)]. In the context of FCA, association rules are generated from closed item sets, where the association rule to be produced relates attributes appearing in the intent of the same concept. A large number of studies focused on how to reduce the number of item sets to explore in order to obtain a complete set of minimal rules [Poelmans et al. (2013b)]. In the scenario

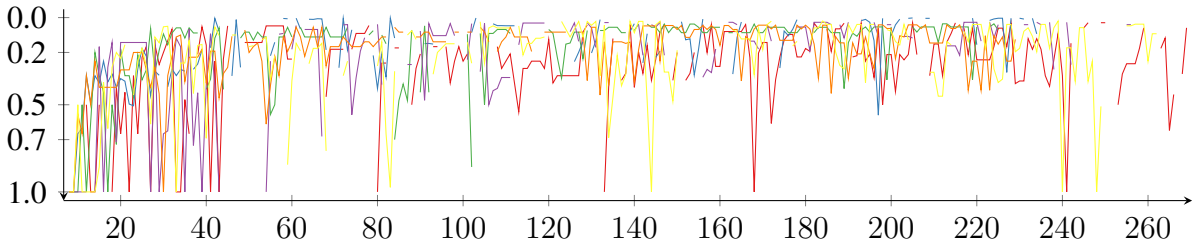


Figure 5.16: Average rank of selected recommendations. The vertical axis represents the score placing at the top the first position.

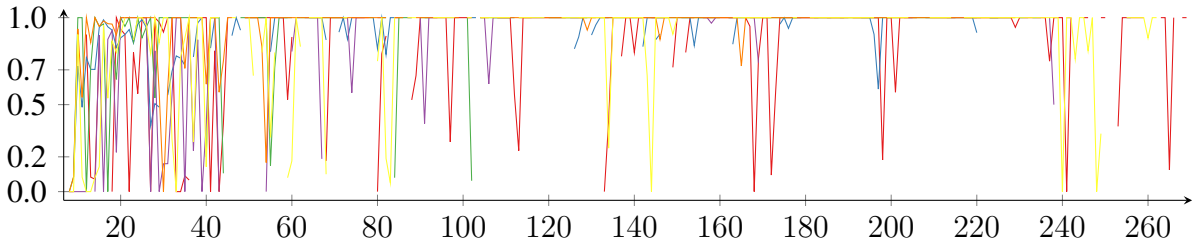


Figure 5.17: Progress of the average relevance score of picked recommendations.

of the present study, where the lattice changes incrementally, generating all the possible association rules would be a waste of resources. The algorithm proposed in the present work is *on demand*, as it only extracts the rules that are relevant for the item to annotate. Our algorithm receives as input an item set, and *retrieves* from the lattice the association rules associated with a relevance score. In other words, we follow an approach unusual with respect to the literature, attacking the ARM problem as an Information Retrieval (IR) one.

The quality and consistency of the resulting annotations are not the subject of the present study, and we did not discussed the interpretation of the Datanode relations with the participants of our experiment. For this reason each user operated on a separate instance of the tool, to reduce the possibility that inconsistent usage of relations would negatively impact the quality of the association rules generated. However, we received feedback that encourages us to better document the Datanode ontology, for example providing cases of the possible uses and misuses of each relation.

In this work we only focused on the relations between input and output within workflow processors. It is possible to extend this approach to also cover relations between data items in other directions (input to input, output to input, etc...).

The FCA component of the Dinowolf Tool is based on an incremental lattice construction algorithm. We plan to integrate a lattice *update* algorithm in order to support modifications to the annotations.

However, the incremental learning of association rules approach presented in this section is independent from both the features of the item to annotate and the nature of the annotations. This opens to the possibility that it could be effectively reused in other scenarios.

5.4 A holistic approach to data cataloguing

In the previous sections we introduced two tools to support (a) data publishers in the selection of the appropriate licence to associate with their data, and (b) data managers in the production of data-centric descriptions of processes relying on the Datanode ontology. Both approaches have the objective to improve the sustainability of the approach we proposed for supporting the task of assessing what policies apply to the output of a complex process, which we called *policy propagation*. Although this assessment is ultimately for the end-user to perform, the objective is to support them by means of an end-to-end solution.

By doing that, we make use of the notion of Supply Chain Management, intended as the activity targeted to optimise networks of suppliers and customers in order to deliver superior value at less cost to the supply chain as a whole. In that setting, a network of interdependent actors mutually and co-operatively work together to improve the flow of materials and information from suppliers to end users. While we use this notion as a metaphor, where the materials are the data and the information the metadata, this comes useful to abstract from the complexity of grounded subproblems, like *data integration*, *metadata storage* or *policy management*, to mention a few. Altogether, they have led us to conclude that, to the best of our knowledge, there is no end-to-end solution for exploitability management today.

In our proposal, such a solution is implemented within a *data cataloguing system* as an essential element of the Data Hub. We propose here a methodology to develop such an end-to-end solution, whose role is to clarify: a) what is the general life-cycle of the data within a Smart City data hub; b) what are the actors involved in such a process; c) what are their goals and tasks; d) what resources are needed, when and how they can be acquired and managed; and e) what operations have to be supported, in order for the exploitability assessment to be performed; finally, f) what are the requirements for the methodology to be applied.

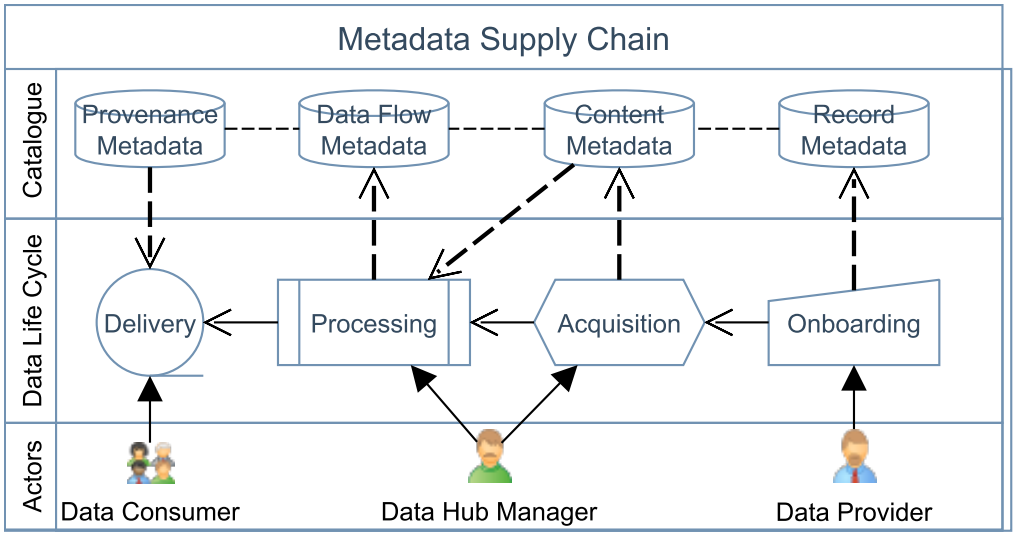


Figure 5.18: Metadata Supply Chain: overview.

The methodology that is introduced in this Section supports what we call "Metadata Supply Chain Management" (MSCM), and is based on a *Data Catalogue*.

Figure 5.18 gives an illustration of the elements of the methodology and their interaction. The primary requirement of our methodology is that a Data Catalogue exists, and is a shared resource on which all the different actors and phases rely. There are 3 types of actors involved in the methodology. A *Data Provider* aims to publish a new data source in order to provide value to the task of a given *Data Consumer*. A *Data Hub Manager* has the role to supervise the infrastructure in terms of configuration, maintenance and monitoring. Our methodology follows a *Data life-cycle*, which comprises four phases:

- **Onboarding**: data sources are registered with the Data Hub;
- **Acquisition**: data are imported in the Data Hub;
- **Processing**: data are processed, manipulated and analysed in order to generate a new dataset, targeted to support some data-relying task;
- **Delivery**: resulting data are delivered to an external system/end-user.

The Metadata Supply Chain Management (MSCM) activity follows the Data Life Cycle in parallel. In the following paragraphs we provide the details of each phase, focusing on:

- the *objectives* that need to be reached;
- the *roles* of the actors in this phase;
- the required *resources* to be managed;
- *operations* to be performed at the different stages;

– what are the *output* resources of each phase; and
– what are the *requirements* that need to be satisfied.

Tables 5.3-5.6 detail each component of each phase in the methodology, and serve as a guide to its implementation in concrete use cases.

Phase 1. Onboarding. The Onboarding phase is dedicated to acquiring and managing information about data sources. When a *Data Provider* wishes to publish a new dataset, the Data Hub has to provide the required facility to do that. From the point of view of the *Data life-cycle*, in this phase the provider registers a new data source (or modifies an existing one) in the Data Catalogue, which is the space where dataset descriptions are managed. The Data Catalogue manages metadata about the data source as a *Catalogue Record*, following the W3C DCAT specification[23]. This description includes details about how the dataset will be populated, and more importantly includes information about ownership (`dc:creator`) and licensing (`dc:license`), as well as attribution statements. The onboarding process requires that the licensor selects a single licence (Requirement 1.1), applicable to whoever exploits the given data source (Requirement 1.2). Licenses are described as set of *policies*, each being a binary association between a deontic component and an action (eg: requirement+attribution, prohibition+commercial_use), according to the definition in Chapter 4 (Requirement 1.4). The range of `dc:license` is meant to be a structured description of a single licence according to the ODRL Ontology, included in a *Licenses Database* (Requirement 1.3).

Phase 2. Acquisition. After onboarding a new data source, the data need to be acquired by the data hub. "Acquiring" means that the data hub is given a means to control the delivery cycle of the data whose awareness was granted through the onboarding phase. It is the role of a *Data Hub Manager* to supervise this process and monitor the acquisition, including implementing the needed strategies for data update and quality control. This activity can be rather complex as it may include automatic and supervised methods, and going into the details of it is out of the scope of this chapter[24]. What is important for us is that this phase should provide a sufficient amount of *metadata* in order to support data processing. *Content Metadata* (see Figure 5.18) refers to topical and structural information that might be established by accessing the actual data[25], to support the configuration of integration strategies by the Data Hub Manager. This phase is based on the assumptions that the data source is actually accessible by the Data Hub (Requirement 2.1) and that acquisition is possible according to

[23]DCAT, `http://www.w3.org/TR/2014/REC-vocab-dcat-20140116/`

[24]For example, data sources could be registered as web accessible resources (via HTTP or FTP), Web APIs, or uploaded files. Methods for acquisition can include collecting resources from external systems or requiring an ingestion API to be exposed.

[25]For example the types of the entities included in the content, the set of attributes, local and global identifiers (and their structure or format), relations and references to external datasets, as well as statistics about them.

Table 5.3: Metadata Supply Chain Phase 1: Onboarding.

Objectives	Obtain information about a data source
Roles	A Data Provider and a Data Hub Manager
Resources	A Data Catalogue, including a Licenses Database, and a data source
Operations	Registration of the data source in the Data Catalogue.
Output	Structured information about the data source in the form of a Catalogue Record.
Requirements *1.1*: The Data Provider associates a single licence to the data source. *1.2*: The licence is granted to whoever exploits the given data source. *1.3*: The licence is described in the Licenses Database. *1.4*: Policies are sets of binary relations between a deontic component (permission, prohibition, requirement) and an action. *1.5*: Policies are referenced by Policy Propagation Rules (PPRs), part of the Licenses Database.	

the data source licence (Requirement 2.2).

Phase 3. Processing. In this phase the data are manipulated in order to fulfil a given task that relies upon them. This activity can be seen as supporting a traditional ETL[26] task. *Content Metadata* include information about the data sources in order to support the configuration of these processes, whether it is an automatic method or a process supervised by the *Data Hub Manager*. However, here we focus on the metadata that the data processing phase must produce in order for the Data Hub to support the user in the assessment of output data policies. From that point of view, a *Data Catalogue* should be capable of collecting plans about the integration processes in order to answer the following question: *when this process is executed, what will the policies attached to its output be?* This can be achieved with the support of Dinowolf, enabling a semi automatic workflow aimed at annotating process models with Datanode relations, as demonstrated in the previous Section 5.3. Metadata about possible *processes* should be collected and stored in the catalogue, in order to allow reasoning on policy propagation, and to attach the required policies to the resulting dataset. Processes can be described as relations between data objects (Requirement 3.1). This is the approach followed by Datanode. Therefore, ETL pipelines can be annotated with *data flow descriptions* as representation of the processes using Datanode, allowing to execute Policy Propagation Rules

[26]Extract, transform, load (ETL), `https://en.wikipedia.org/wiki/Extract,_transform,_load`.

Table 5.4: Metadata Supply Chain Phase 2: Acquisition.

Objectives	Access the data source and collection of the related data.
Roles	The Data Hub Manager supervises and monitors the relevant procedures.
Resources	A Catalogue Record, containing information about how to access the data.
Operations	Collection of the data, inspection and eventually storage in a staging environment.
Output	Content Metadata, ready to be exploted by the required processes.
Requirements *2.1*: The data source is accessible. *2.2*: Acquisition is performed by respecting the data source License.	

(PPRs) and determine what policies can be attached to the output of each process [Daga et al. (2015a)]. In a general case, the *Data Hub Manager* is responsible for providing such information, as well as assessing that the processing itself is made respecting the policies of the data sources (Requirement 3.2). Moreover, these metadata should provide an abstract representation of the process so that, once combined with the actual input (a given data catalogue record and content metadata), it would be possible to generate the relevant policies. In other words, a given data flow description should be valid for all possible executions of a process (Requirement 3.3).

Phase 4. [Metadata Supply Chain Phase 4: Delivery.]Metadata Supply Chain Phase 4: Delivery.. In this phase data are delivered to the end user or application. The Data Catalogue provides the required metadata to be distributed alongside the process output. Delivered data should include provenance information such as ownership, attribution statement and policies (permissions, requirements, prohibitions). Delivered metadata should be included in the provenance information (Requirement 4.2), in order to support the user in assessing the data exploitability for the task at hand[27]. Once the metadata reach the end-user, the exploitability task is indeed reduced to the assessment of the compatibility between the actions performed by the user's application and the policies attached to the datasets, with an approach similar to the one presented in [Governatori et al.

[27]It is worth noting that the actual assessment of compatibility between the user's task and the policies of the output data is not part of this methodology, and is left to the end user.

Table 5.5: Metadata Supply Chain Phase 3: Processing.

Objectives	Obtain a description of the ETL process suitable to reason on policies propagation.
Roles	The Data Hub Manager to configure the processes and produce descriptions of the data flows.
Resources	A Catalogue Record linked to Content Metadata. Processing will need to exploit the former or the latter, on a case by case basis.
Operations	Processes must be described as networks of data objects relying on the Datanode ontology.
Output	Data flow descriptions to be registered in the Data Catalogue.
Requirements *3.1*: Processes can be described as data flows with Datanode. *3.2*: ETL processes do not violate the licence of the source. *3.3*: Process executions do not influence policies propagation.	

(2013b)], for example using the SPIN-DLE reasoner[28].

5.4.1 Evaluation

Our hypothesis is that an *end-to-end* solution for exploitability assessment can be developed by using state-of-the-art techniques through the implementation of the methodology introduced so far. We now validate this statement by describing the solution developed in the MK Data Hub, following the scenario introduced in Section 5.1.

Figure 5.19 illustrates the components and their role in the data and metadata life-cycle of the MK Data Hub. The Onboarding phase is the initial step of our methodology, and it is supported by providing an *input interface* to *Data Providers*, implemented as a Data Hub Portal page and a Web API. Following our sample use case, some data sources are registered in the Data Catalogue. They are *Air Quality and Moisture Sensors* in the Bletchley area, the *Flickr API* (including a number of images annotated with geocoordinates associated with the ward), the *UK Food Estanblishments Info and Ratings API*, as well as topographical information exposed by the *Ordnance Survey* and *statistics* from the Milton Keynes Council. Each one of these data sources have a single licence

[28]SPIN-DLE, `http://spin.nicta.org.au/spindle/index.html`

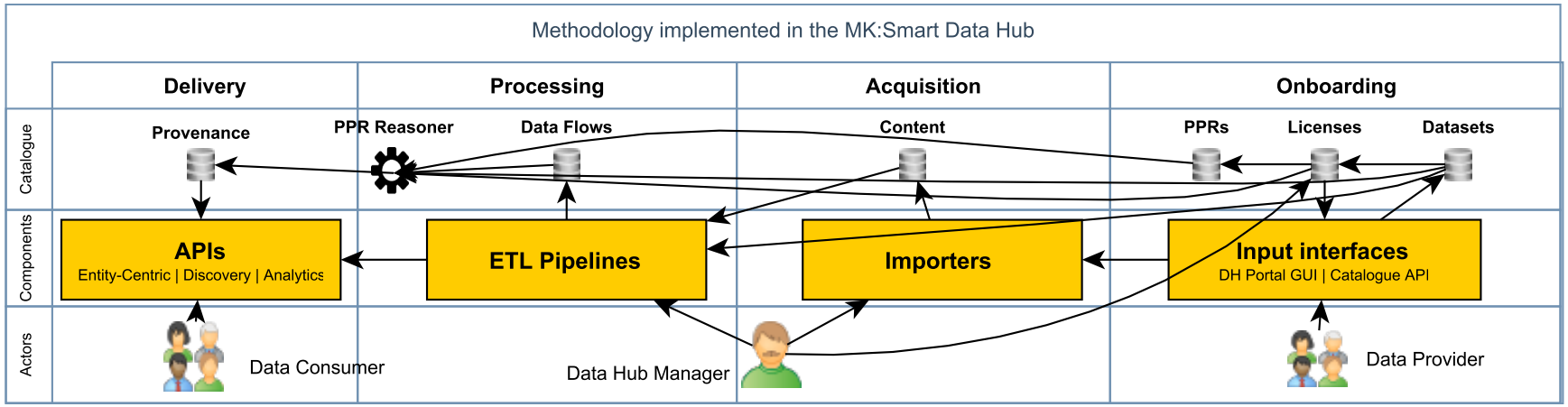


Figure 5.19: MK Data Hub overview. The figure shows the phases of the methodologies within the context of the MK Data Hub. The *Data Catalogue* is the component responsible for managing the *Metadata Supply Chain*, interacting with the other components of the system. On the right side of the image, *Data Providers* to register new data sources. A *Data Hub Manager* is responsible for the description of licenses, and supervises the activity of *importers* and ETL *pipelines*, including the curation of data flow descriptions (*Data Flows*) and policy propagation rules (*PPRs*). *Data Consumers* invoke APIs and associated *Provenance* information is provided from the *Data Catalogue*, exploiting a *PPR Reasoner* that relies on *Data Flows* descriptions and *PPRs*.

Table 5.6: Delivery

Objectives	Deliver the set of policies associated with the data as part of the provenance information.
Roles	The Data Consumer.
Resources	Catalogue Record, Data flow metadata, Policy Propagation Rules base.
Operations	Reason on PPRs given the data flow description and the rule base.
Output	Set of policies attached as part of the provenance information of the returned data.
Requirements *4.1*: Data flow descriptions and licence policies enable reasoning on Policy Propagation Rules. *4.2*: End-user access method includes provenance information.	

associated (R 1.1), applicable to whoever makes use of the data (R 1.2), and described in RDF/ODRL in a *Licenses Database* (R 1.3, see also Figure 5.19). For example, the metadata about the *Water Level Gauge - Bletchley - E21505* data source is one of the relevant data sources for the area. Figure 5.1 shows the Data Catalogue record as presented in the MK Data Hub web portal. As shown in Listing 5.3, the related description includes a reference to the *Open Government License*, described following Requirement R 1.4 (see Listing 5.4). These policies have related PPRs in the Licenses Database (R 1.5). It is the role of the Data Hub Manager to provide the necessary descriptions in the licence Database.

Listing 5.3: Dataset: Water Level Gauge - Bletchley - E21505: RDF description.

```
:water-level-gauge-bletchley-e21505
 a dcat:Dataset ;
 dc:title "Water Level Gauge - Bletchley - E21505" ;
 mks:owner "Environment Agency" ;
 mks:policy policy:open-government-license ;
...
```

Listing 5.4: Open Government License: policy set.

```
:open-government-license
```

```
odrl:permission [
 a odrl:Permission;
 odrl:action
   odrl:derive,
   odrl:distribute,
   ldr:extraction,
   odrl:reproduce,
   odrl:read,
   ldr:reutilization;
 odrl:duty [
   a odrl:Duty;
   odrl:action odrl:attachPolicy,
   odrl:attribute
   ]
 ] .
```

Data sources like the *Flickr API* come with peculiar terms and conditions[29] (Listing 5.5). Some of them refer to the usage of the API, others to the assets the data are describing (like Flickr images). In these cases the Data Hub Manager limits the descriptions to the policies that are applicable to the accessed data, and describe them in the Licenses database. The description always include a reference to the document from which the policies have been extracted.

Listing 5.5: Flickr TOS.

```
mks:flickrtos
  odrl:prohibition [
    a odrl:Prohibition;
    odrl:action
      odrl:sell,
      odrl:sublicense,
      cc:CommercialUse
  ];
  odrl:duty [
    a odrl:Duty;
    odrl:action odrl:attribute
  ];
  mks:attributionStatement "This product uses the Flickr API but is not
     endorsed or certified by Flickr." ;
  dct:source <https://www.flickr.com/services/api/tos/> .
```

This applies also to other sources taken into consideration, like the *UK Food Establishments Info and Ratings*[30] as well as statistics from the MK Council, which come with an Open Government License. The result of this phase is a set of *Catalogue Records* supporting all the requirements of

[29]Flickr: `https://www.flickr.com/services/api/tos/`

[30]The dataset includes a snapshot of the food hygiene rating data published at `http://www.food.gov.uk/ratings`.

the initial phase of our methodology.

The Acquisition phase is the stage of the methodology that covers the execution of the processes required to populate the dataset from the sources, assuming they are accessible (R 2.1). This can be achieved in different ways in the MK Data Hub. For each type of source the data cataloguing system implements a dedicated *metadata extractor* with the objective to complement the *Dataset Record* with more metadata for supporting the data processing. For example, air quality and soil moisture sensors push regular streams of data in the Data Hub. The Flickr API is invoked on demand and information stored at query time in temporary datasets. During these processes, metadata about the geolocalisation of the related items are extracted and stored in the Data Catalogue. *Content Metadata* include the location of the Flickr images, while geocoordinates of the sensors are part of the *Dataset Record*. It is the responsibility of the Data Hub Manager to verify that the acquisition of the data is possible without violating the sources' terms of use (R 2.2). In the Processing phase, data are extracted, transformed and loaded (ETL) in datasets using dedicated *pipelines*. Each pipeline performs a number of operations on the data sources in order to select the relevant information and transform it in a format suitable for the task at hand. Listing 5.6 shows the description of the processing pipeline of a file data source from Milton Keynes Council, as configured by the Data Hub Manager. The file is downloaded from the remote location and a copy is stored locally in a staging area (see also Figure 5.19). The content is then transformed into RDF using the CSV2RDF approach[31]. After that, a SPARQL query remodels the data applying the W3C Datacube Vocabulary[32] data model. These data are accessed by a SPARQL query, which selects a relevant portion of the data for the task at hand.

Listing 5.6: Processing pipeline for a CSV file.

```
:input a dn:Datanode;
  mks:format mks:csv;
  dn:hasCopy [
   dn:refactoredInto [
    mks:format mks:rdf;
    dn:usesSchema csvOntology: ;
    dn:remodelledInto [
      dn:usesSchema qb: ;
     dn:hasSelection :output ]]] .
```

The descriptions of the data flows are provided a single time by the Data Hub Manager following Requirement R 3.1. In this activity, the Data Hub Manager verifies that the ETL process is compliant

[31] http://www.w3.org/TR/csv2rdf/

[32] http://www.w3.org/TR/vocab-data-cube/

with the source licence (R 3.2). This model represents the process in an abstract way, and it is agnostic with respect to the actual input (R 3.3).

The Data Hub exposes a number of APIs to access the data in various forms. For example, sensor data can be extracted as streams by providing temporal constraints. The Entity Centric API is a specialised service for data discovery that aggregates information summaries from several datasets about a given entity. In our running examples, a *Data Consumer* requests information about a location in Milton Keynes, in the form of geocoordinates: `51.998,-0.7436`. The PPR Reasoner will be queried providing the actual input as a specific dataset in the catalogue, according to the user's query (R 4.1).

Listing 5.7: Policy Propagation Rules.

$$propagates(dn : remodelledTo, duty\ cc : ShareAlike)$$
$$propagates(dn : hasSelection, duty\ cc : ShareAlike)$$
$$propagates(dn : hasCopy, duty\ cc : ShareAlike)$$

The dataflow description will be complemented by the related dataset record metadata and associated policies from the licenses database. Listing 5.7 shows a subset of the rules that are activated in relation to the dataflow (Listing 5.6) and policies set (Listing 5.4). The propagated policies are displayed in Listing 5.8.

Listing 5.8: Policies associated with the returned data processed from the original Milton Keynes council CSV file.

```
[] a dn:Datanode ;
    odrl:duty [odrl:action odrl:attachPolicy, odrl:attribute]
```

The output includes an aggregated view of items related to that geolocation as well as provenance information for each one of them, including the policies relevant to assess the exploitability of each item (R 4.2).

5.4.2 Discussion

We described how the MK:Smart Data Catalogue supports the methodology proposed so far. Table 5.7 summarises the assumptions upon which the methodology relies. In this section we are going to discuss *to what extent* the assumptions are valid for the MK Data Hub.

Table 5.7: Metadata Supply Chain: Assumptions.

Id	Assumption
1.1	The Data Provider associates a single licence to the data source.
1.2	The licence is granted to whoever exploits the given data source.
1.3	The licence is described in the Licenses Database.
1.4	Policies are sets of binary relations between a deontic component (permission, prohibition, requirement) and an action.
1.5	Policies are referenced by Policy Propagation Rules (PPRs), part of the Licenses Database.
2.1	The data source is accessible.
2.2	Acquisition is performed by respecting the data source License.
3.1	Processes can be described as data flows with Datanode.
3.2	ETL processes do not violate the licence of the source.
3.3	Process executions do not influence policies propagation.
4.1	Data flow descriptions and licence policies enable reasoning on Policy Propagation Rules.
4.2	End-user access method includes provenance information.

Assumption 1.1 — The Data Provider associates a single licence to the data source. Each Dataset is supposed to be annotated with a *single* license. The MK Data Hub contains today[33] 754 datasets, of which 228 are openly accessible. All of them specify a single license. This assumptions is fully valid for existing datasets. However we can expect cases where the licence can change depending on the type of user or the context of applications. This is case of data from the BBC Weather Service, which terms and conditions[34] for commercial and non commercial use are different, and are also specified in different documents. While we do not support complex policies at the moment, we could deal with it by user profiling (with a commercial or non commercial account), or by including a taxonomy of usage contexts to consider separately, thus obtaining multiple policy sets depending on the usage context. Delivered metadata could include multiple policy sets associated with the related contextual information.

Assumption 1.2 — The licence is granted to whoever exploits the given data source. The methodology assumes the licence of the data source to refer to any possible user having access to the data source. However, we can imagine situations in which the terms of use may vary depending on different kind of users, because of private agreements between the parties. While the MK Data Hub does not support this facility, it is possible to envisage an extension of the methodology in which the licence Database contains associations between licenses and (classes of) users, thus enabling the configuration of the PPR reasoner in order to select the relevant licence between the set of possible ones. This can be supported particularly because the licence is part of the input of the PPR reasoner, together with dataflow description.

Assumption 1.3 — The licence is described in the Licenses Database. In the MK Data Catalogue, the number of datasets that do not specify a licence is 235 ("Other" in Table 5.8). There can be many reasons for that. In some cases Data providers do not want (yet) to redistribute the content of the dataset, and rely on the MK Datahub solely for their own applications. Sometimes the intended licence is not present in the current selection of licenses. When this happens, the user can contact the Data Hub Manager and discuss her specific requirement. In the future, we plan to allow the users to create entirely customised policies to be associated with their data, as supervised by the Data Hub Manager, and in cooperation with the legal team of the Data Hub. Table 5.8 summarises the licenses currently used.

[33] October 2017.

[34] `http://www.bbc.co.uk/terms/`

Table 5.8: Licenses and their use.

N	licence
48	Open Database License (ODbL) v1.0
89	Open Government License
235	Other
20	Netatmo API Terms of use
1	Terms and conditions for information and services at food.gov.uk/ratings
2	Flickr APIs Terms of Use
4	Creative Commons Attribution-ShareAlike 4.0 International
4	Koubachi Platform Terms of Service
348	Creative Commons Attribution 4.0 International
3	OS Open Data License

Assumption 1.4 — Policies are sets of binary relations between a deontic component (permission, prohibition, requirement) and an action. Policies can have a very diverse structures, including composite constraints involving actions, classes of users, conjunctions, disjunctions, etc. While these can be represented in ODRL, in this work we only focused on policies having a flat representation, i.e. a binary association between a deontic component and an action. Policy Propagation Rules treat the policy as an atom that can or cannot propagate through relations between datanodes. For this reason, the actual structure of the policy does not affect the behaviour of the rule, and we can extend our framework to also work on more complex ones. This would have an impact on the life cycle of policies and licenses definition, which should be extended to also manage these kinds of policies, when necessary. At the moment these policies are not represented in the Licenses Database. However, we performed an informal evaluation of this aspect using the RDF licence Database, which contains a number of licenses expressed as RDF/ODRL. We observed that all the RDF/ODRL policies expressed in the database can be reduced to sets of binary associations between a deontic component and an action, thus supporting this assumption[35].

Assumption 1.5 — Policies are referenced by Policy Propagation Rules (PPRs), part of the Licenses Database. In order for the process to be successful, all policies used to describe licenses in the Licence Database need to be referenced appropriately by Policy Propagation Rules. Policies

[35]However, we did not performed a validation of the accuracy of the RDF Licenses Database

introduced by new licenses should be also included in the set of rules. In [Daga et al. (2015a)], a methodology to manage a knowledge base of policies propagation rules is presented, and we rely on that approach to manage the evolution of the rule base in order to guarantee that any policy in the licenses database is properly represented by PPRs.

Assumption 2.1 — The data source is accessible. *Data Catalogues* are conceived as metadata repositories, which act as registries of existing datasets. In our methodology, ETL processes rely on Content Metadata that is generated by inspecting the data source, thus establishing a dependency between the Dataflow description and the access of the actual data. While it is obvious that derived datasets cannot be generated without accessing the input data source, we can envisage situations in which data flow descriptions can be generated with no need to access the data source. One example is when the structure of the data conforms to an existing standard and the process itself is agnostic to the population of the dataset. In these cases, process executions can be simulated by running the PPR Reasoner with the related data source (metadata) as input.

Assumptions 2.2 — Acquisition is performed by respecting the data source License. and 3.2 — ETL processes do not violate the licence of the source. The Data Hub Manager has the responsibility to respect the terms and conditions on the data access method as well as the ETL procedures involved. This assessment can be performed by inspecting the data source licenses. While currently the MK Data Hub does not support ETL processes involving multiple datasets, these cases can be also supported by relying on the licenses compatibility approach. The need of setting up dataflow descriptions guarantees that there exists one operation/phase under which this assessment will be performed.

Assumption 3.1 — Processes can be described as data flows with Datanode. The primary implication is that Datanode is capable of describing the data flow. Datanode is an evolving component and it can be extended by adding new relations in the ontology. This can also evolve the Policy Propagation Rules database, following the method described in [Daga et al. (2015a)]. For this reason, we can assume that Datanode will have enough expressivity to cover existing dataflows. The generation of the policy set to attach to the output is performed at runtime. This method allows for process descriptions to be reusable between different executions. However this implies that processes need to be careful not to change the implications of the policies at runtime. For example, if some policy applies to a specific section of the data, different runtime executions might have different policies depending on the selected data. This aspect is not currently supported and processes are designed in order to be agnostic to runtime information (user's input). Without this

assumption, process executions should be able to provide fine grained traces (e.g.: logs) that could be then transformed in Datanode graphs. This could be an interesting future work to experiment with.

Assumption 3.3 — Process executions do not influence policies propagation. This assumption is an implication of Assumption 1.1. If policies are attached to the whole dataset, different executions of the same process will always refer to the same set of policies. Dataflow descriptions are based on the operations performed by the ETL process on hypothetical inputs. At runtime, the concrete data source is selected, thus the set of policies of the related license. This is not necessarily always true. As a negative example, we can imagine a dataset including policies attached at instance level. The records referring to the current year cannot be used for commercial purposes, while data about the past years are of public domain. Depending on the input of the query, a process might or might not select restricted data, thus changing at runtime the information required to assess policies propagation. We solve this problem by slicing the data source in different *Catalogue Record*s, with different licenses.

Assumption 4.1 — Data flow descriptions and licence policies enable reasoning on Policy Propagation Rules. Following the approach presented in previous chapters, and given the *Assumptions 1.5* and *3.1*, the PP Reasoner will have sufficient information to reason on policies propagation.

Assumption 4.2 — End-user access method includes provenance information. Finally, the methodology assumes that the user has access to some metadata (Provenance information). The user's task need to be expressible in terms of ODRL policies, thus enabling reasoning on policies compatibility. However, while this assessment is part of an early analysis, when the user wants to assess whether a given dataset is eligible to be adopted, we expect this assessment to be performed manually, on a case by case basis. We plan to extend the MK Data Hub Portal to also support a friendly user interface that users can exploit to validate the policies with respect to her requirements.

5.5 Conclusions

In this chapter we focused on the support we can give to users in the task of deciding about the propagation of policies, and aimed at answering $R.Q.\ 3$. We took Smart Cities as the reference scenario, and a Data Hub as an infrastructure where various stakeholders collaborate in order to fulfil their data relying requirements. In a Data Hub, data publishers, data managers and data consumers

operate together through an infrastructure that mediates the different tasks at the technical level, but that is agnostic with respect to the rights associated with the components involved. Within this scenario, we verified hypothesis $H.\ 4$, and demonstrated how a semantic representation of licences and processes can aid the support required for the acquisition of the required knowledge components. In particular, we shown how a semantic description of licenses, while being necessary to reason on policy propagation, can also support users in the task of assigning the appropriate licence to their content. Similarly, we demonstrated that it is possible to support the user in the acquisition of the required process knowledge by annotating existing workflow description without the need of a pre-existing repository of annotated resources, by exploiting existing process models.

In Section 5.2 the focus has been put on data publishers that need to question themselves and choose the licence that better suits them from the ones available in a Licence Catalogue. In Section 5.3, we faced the issue of supporting data managers in the production of Datanode description of processes, by relying on the features of previously existing process models. Finally, in Section 5.4 we designed a methodology for supporting data exploitability as the assessment of the compatibility of data policies with the task at hand, and validated this methodology through its implementation within the MK Data Hub platform.

The MK Data Hub indeed supports the methodology proposed in this article. A *Data Provider* registers a dataset in the Data Hub, and can indicate a licence from the ones available in the Licenses Database, containing a set of licenses described in RDF/ODRL, using the approach presented in Section 5.2. These policies are mapped to Policy Propagation Rules following the approach described in Chapter 4. During the import phase, *Content Metadata* is extracted, assuming that all the relevant information to setup data integration strategies is available. A supervised process produces: a) a configuration for the processes to be executed and b) a description on the process with Datanode, with the support of the method and tool presented in Section 5.3. Since policies and data flows are described according to the process in Chapter 4, they enable a *PPR Reasoner* to execute Policies Propagation Rules in relation to the process dataflow description and to generate the part of *Provenance Metadata* to be attached to the result of the call to the ECAPI. According to this process, the end user will have enough information to select the appropriate dataset to fulfil her task, according to whether her requirements match the policies associated with the dataset descriptions, which are therefore supporting exploitability assessment.

Chapter 6

Evaluation in the Context of a Smart City Data Hub

In the previous chapter, we showed how users can be supported on the task of policy propagation. We developed a Policy Propagation Reasoner (PP Reasoner) and various associated tools to support data managers and users in developing the needed *knowledge components* and use them to make decisions about the policies associated with derived datasets. By means of a semantic representation of policies and of data flows, the system combines OWL reasoning and Policy Propagation Rules (PPR) to compute the policies associated with any data node involved in the process.

However, in our previous work we focused on the technical feasibility of the approach in terms of knowledge acquisition and management (Chapter 4, Sections 4.1-4.2), scalability of the reasoner (Chapter 4, Section 4.3), and applicability in an end-to-end user scenario (Chapter 5, Section 5.4)). So far, we relied on the assumption that indeed it makes sense to reason on policy propagation and in particular that a semantic representation of the actions of a process and of the involved policies is enough to obtain accurate answers. However, this assumption requires an investigation of its own. Therefore, we performed a user experiment in order to evaluate the feasibility of policy propagation as a solvable problem and the hypotheses behind the development of the system, by relying both on quantitative and qualitative data analysis methods, particularly the Grounded Theory (GT) approach[1], in a comparison between the automatic process and a manual one performed by people with the typical skill set of data consumers, processors and publishers who would be carrying out this task in a realistic context.

The participants were confronted with the problem of deciding what policies need to be taken into account when using a dataset that was derived from a complex process reusing licensed data

[1] https://en.wikipedia.org/wiki/Grounded_theory

sources. Their decisions were then compared with the ones of our system, and insights into its expected behaviour were acquired as well as observations about its accuracy and utility. In the next section, we summarise the PP Reasoner and the knowledge bases it relies upon. Section 6.2 describes the setup of the user study, the methodological criteria, and how the required resources were acquired and developed. We discuss the feedback received from the participants of the user study in Section 6.3, before going into the details of the results in Section 6.4 and discussing them in Section 6.5. Section 6.6 summarises the contributions of this study.

6.1 Description of the system at a glance

The role of the PP Reasoner is to support users in the assessment of the impact of input data policies on the exploitation of the output data of processes and workflows. Consider the case where Food rating data released by a trusted authority under a licence that prohibits distribution is used alongside public data about city roads in order to assess the best Machine Learning approach, among several options, to employ for the prediction of good quality restaurants. This task would produce two types of outputs: (a) a set of datasets about roads labelled with the expected food quality rating; and (b) a set of datasets including details about the performance of each one of the algorithms tested. While the prohibition of distribution should be taken into account when using the former datasets, the same constraint would not apply to the latter.

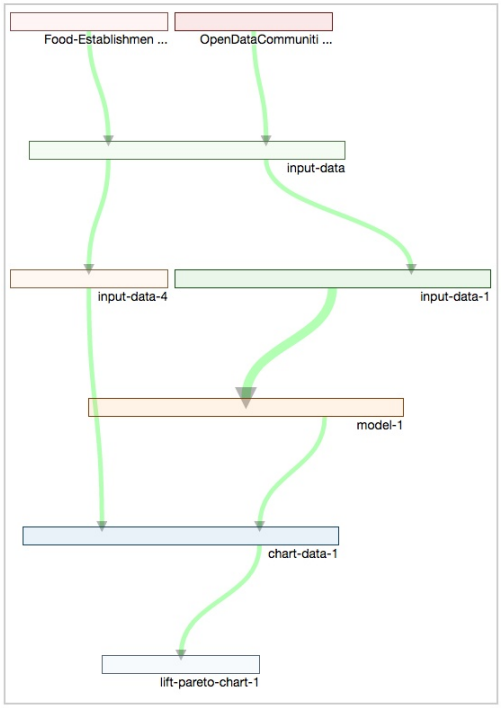
The system is designed to work with a set of reference knowledge bases:

- *Data Catalogue.* Provides general metadata for datasets, including the link to the associated policy set (licence, Terms and Conditions, and so forth).
- *Licence Catalogue.* Includes the set of licences represented using the ODRL Ontology.
- *Process Catalogue.* Defines the set of processes represented using the Datanode Ontology.
- *Policy Propagation Rules (PPRs).* A rule base developed and managed as described in [Daga et al. (2015a)]. Rules have the form of a connection between an atomic policy and a relation that is supposed to propagate it. For instance, `propagates(dn:cleanedInto, odrl:permission cc:DerivativeWorks)` instructs the reasoner to propagate `odrl:permission cc:DerivativeWorks` whenever a data item is `dn:cleanedInto` another, so that the cleaned item would also have the given policy.

The system was developed using the OWL reasoner of Apache Jena[2] in conjunction with a

[2]Apache Jena: `http://jena.apache.org`

SPIN[3] rule engine. By relying on ODRL policies, Datanode graphs, and PPRs, the system computes the propagated policies for each node in the process graph[4]. The resulting RDF graph can be queried to obtain the policies of the output datanode. Moreover, the system can generate an explanation of the decision like the one in Figure 6.1, which traces the lineage of a given policy and highlights the arcs that would propagate it or block it[5].



Policy	duty Attribution		
Output	lift-pareto-chart-1		
From	Property	To	
Food-Establishmen ...	isPartOf	input-data	
OpenDataCommuniti ...	isPartOf	input-data	
input-data	duplicate	input-data-1	
input-data	duplicate	input-data-4	
input-data-1	hasInterpretation	model-1	
input-data-1	hasComputation	model-1	
input-data-4	combinedIn	chart-data-1	
model-1	combinedIn	chart-data-1	
chart-data-1	remodelledTo	lift-pareto-chart-1	

Figure 6.1: Explanation: propagation trace.

6.2 User Study Methodology

The objective of the present study is to evaluate to what extent it is possible to support users on taking decisions upon the propagation of policies, and whether it is useful using the system outlined above. We assess this in two ways: 1) by comparing the decisions of the system with the ones performed manually in a quantitative analysis of the system's *accuracy*, and 2) by discussing the

[3]SPIN: `http://spinrdf.org/`

[4]More details about the implementation of the reasoner can be found in [Daga et al. (2016a, 2015a, pear)]

[5]The system's objective is to compute the set of propagated policies and it does not check the consistency of the policies. This could be done for the output set by relying on state of the art deontic reasoners like the one used in [Governatori et al. (2013b)]. In the present work we are only interested in computing policy propagation in relation to the actions performed in different processes.

issues raised in disagreements with a qualitative analysis of the users' decision process. We further assess the value to users of the automatic support by asking the participants a set of questions concerning the experience of reasoning upon the policies and how they relate to processes, through a feedback questionnaire. In this section we illustrate the methodology employed in the study, including the *design* of the experiment, the criteria followed in the *sampling* of the users and the scenarios, and for data *collection* and *analysis*.

Design. The experiment simulated a set of scenarios in which a Data Hub manager needs to take a decision about which policies need to be associated with a dataset derived from a complex data manipulation process performed by the Data Hub infrastructure. We provided the participants with the same knowledge as the one used by the system, asking them to perform a number of decisions about policy propagation in reference scenarios, which we called *data journeys*. On each data journey, input datasets and the associated licences were presented, as well as a formalised representation of the process. Users were asked to make decisions about which ones of the policies derivable from the input licences should also be applied to the output data. We asked them only to decide whether a policy would propagate to the output, ignoring whether the process itself would violate the policies, or whether the propagated policies would be consistent with each other. Users were especially asked to compare their choices with the system and discuss potential disagreements. The study was conducted with the support of a Web tool we developed, which guided the participants in the process.

A session started with an introductory phase, where the participants were given a short presentation about the Data Hub and the task they were going to perform, exemplified by a tutorial data journey. Then, the participants were left to face two *data journeys* involving real data. At the end of the sessions, users completed a feedback questionnaire individually.

A single data journey was structured as follows:

1. **Understand the process.** Participants were asked to become familiar with the data process, described with the Rapid Miner tool[6].
2. **Understand the input datasets.** In this phase, the tool listed the dataset(s) selected to be the input source(s) and, for each dataset, the set of permissions, prohibitions or duties associated. Participants were asked to check their understanding of the nature of the actions that are mentioned (also documented by the tool according to their specification in the related semantic

[6]In particular, they were suggested to answer the following questions: what is the nature of the input data? What is the purpose of the process? What are the intermediary steps of the process? What is the nature of the ouput data?

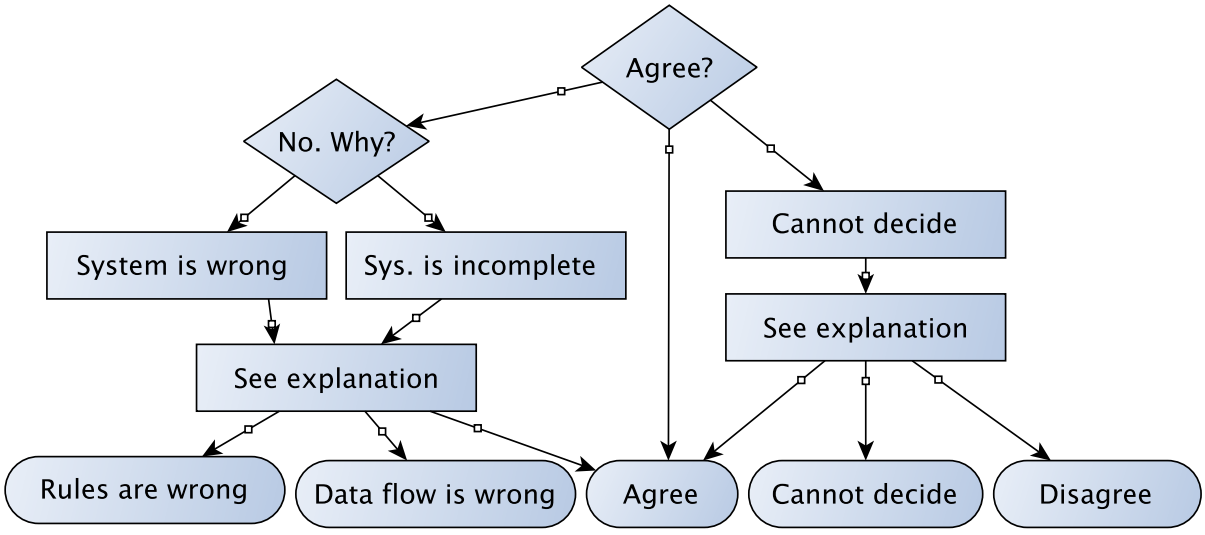


Figure 6.2: The decisions of the participants are compared with the ones of the system.

web ontology - often ODRL but also CC[7], LDR[8]).

3. **Indicate what policies shall propagate to the output.** Users are asked to indicate whether each one of them should be applied to the output, in the form of Likert-scale questions: *(-2) Certainly not*, *(-1) Probably not*, *(0) I don't know*, *(+1) Probably yes*, or *(+2) Certainly yes*.
4. **Compare with the automatic reasoner.** This phase is summarised in Figure 6.2. The choices of the users are compared with the ones of the system, and conflicting ones are highlighted. It is worth noting that, while the users were requested to express an opinion with some degree of uncertainty, the system would always return a boolean answer: the policy is propagated or it is not. The journey terminated after all disagreements were discussed.

In the last phase, users are requested to *resolve* each one of the disagreements, depending on the following possible situations:

1. *The system is wrong.* The tool does propagate the policy to the output, but the users think it should not.
2. *The system is incomplete.* The tool does not propagate the policy to the output, but users think it should.
3. The users could not decide (answer 0).

In all cases, a *propagation trace* was proposed to the users to explain the system's decision (see Figure 6.1). Users could either: a) agree with the representation, but indicate that some relations should

[7]Creative Commons Rights Expression Language: `https://creativecommons.org/ns`
[8]Linked Data Rights: `http://oeg-dev.dia.fi.upm.es/licensius/static/ldr/`

behave differently (i.e. that `dn:hasCopy` should propagate the duty of `cc:Attribution`); b) disagree on the way the data flow was represented, and indicate how it should be; c) change their opinion after seeing the explanation and agree with the system. In cases where users could not decide, they were asked to justify why they believed they could not decide. We also gave them the possibility to abort the task, showing why a decision could not be made. In all cases, we asked them to compare their decision with the result of the system and to examine the explanation (*propagation trace*, see Figure 6.1), in order to collect insights into what to fix in the system.

Sampling: participants and scenarios. Ten participants were involved, selected among researchers and PhD students in our university, all having a background that includes some data analytics skills. The absence of a specific legal expertise in the study participants is intended. We evaluate the system with users who would typically perform such tasks. These people are developers, data scientists and practitioners who would process, reuse and republish data. Realistically we cannot assume them to have legal knowledge. To improve the quality of the decisions, we grouped the participants in teams of two persons, asking them to be in agreement before taking action. Moreover, we introduced one intentional anomaly in the system, to check that users were paying enough attention during the study. We will refer to the five teams as follows: $MAPI$, $ILAN$, $CAAN$, $ALPA$, and $NIFR$.

Much effort was allocated to setting up realistic scenarios, comprising real data sources used in conjunction with real processes. The MK:Smart project has collected a large quantity of data sources about the city of Milton Keynes [d'Aquin et al. (2015a)]. The datasets used in the study are real data sources selected from the MK Data Hub data catalogue[9]. However, Data Hubs also generate new data sources out of existing ones as a result of analytics. In order to select realistic workflows to be used in our experiment, we searched for pre-existing processes, instead of designing ad-hoc resources. We focused on data analytics processes developed outside the MK Smart Data Hub with a significant degree of formalisation. Rapid Miner[10] is a popular tool that supports users in the design of articulated processes by means of a graphical user interface, making it a good candidate for the selection of our exemplary processes. Therefore, we explored the open source projects available on GitHub[11] searching for Rapid Miner process files. We selected *five* workflows representative of common data-intensive tasks that could be applied to MK:Smart data. We designed five scenarios by associating them with real datasets from the MK Data Hub. From these associations, the *data*

[9]MK Data Hub: `http://datahub.mksmart.org`
[10]Rapid Miner: `https://rapidminer.com/`
[11]GitHub: `https://github.com/`

journeys listed in Table 6.1 were designed[12]. Each data journey has some exemplary characteristics. $SCRAPE$ refer to the very common expedient of crawling data out of web resources in order to set up a textual corpus. The $FOOD$ journey is the scenario already mentioned in Section 6.1, where a data source is used to evaluate a Machine Learning approach. $CLOUD$ refers to the extraction of textual data from microblogs. There are many kinds of statistical operations that can be performed on data, AVG is about the calculation of a *moving average*. A large part of the effort of applications relying on sensors is put in data preparation. This aspect is well reflected in the $CLEAN$ data journey.

Each team was given 2 data journeys, and each data journey two teams. Later, we will compare the system twice on each scenario, and the teams between each other, in the agreement analysis. The data journeys were allocated following a latin square, thus avoiding to assign two tasks to the same two groups. We chose scenarios that were (a) complex enough, (b) feasible within 2 hours (so people would not be too fatigued), and (c) diverse enough in terms of use case (and type of operations performed). Although we cannot and do not formally claim for those scenarios to be a representative set of cases (because we cannot have all the cases), we can safely assume that they are ecologically valid. Also, the licenses are different in each scenario, covering a diverse range of policies: 15 permissions, 17 prohibition and 8 duties, selected from the 119 policies in the system. Overall, the experiment concerned 77 decisions including 40 policies, as often a policy was present in more than one scenario.

Data collection. During the experiment, we acquired three types of data: a) the decisions taken by the teams and the system, registered by the tool; b) a record of the motivations behind the decisions, in particular about disagreements with the system and borderline cases; and c) a feedback about the general difficulty of the task and the perceived user value of our system, obtained through a questionnaire including nine closed-ended leading questions and one single-choice question (see Table 6.2 and Figure 6.3). We attended the study as supervisor providing support in the overall process - for example when the users needed clarification on the semantics of workflow actions or on the usage of the tool, but avoiding to influence their opinion on whether a policy ought to be propagated or not. Sessions' audio was recorded as well as the operations performed on the screen, preserving the discussions and small talks occurred motivating the rationales behind users' decisions.

[12]The scenarios were build upon the analysis of the process, and considering the dataset selected as *equivalent* to the one for which the process was originally designed. As a consequence, it is possible that the process would not work as-is with our data. However, this is not a problem as its steps could potentially be applied with the appropriate functional modifications. In any case, none of the scenarios were actually executed.

Table 6.1: Data Journeys (a,b).

(a) SCRAPE

SCRAPE	**Milton Keynes Websites Scraper.**
The content of Websites about Milton Keynes is downloaded. Each web page is processed in order to select only the relevant part of the content. After each iteration the resulting text is appended to a dataset. The resulting dataset is modified before being saved in the local data storage.	
Datasets	Milton Keynes Council Website (UK OGL 2.0), MK50 Website (All rights reserved), Wikipedia pages about Milton Keynes (CC-By-SA 3.0)
Policies	Permissions: reutilization, Reproduction, Distribution, DerivativeWorks. Prohibitions: Distribution, IPRRight, Reproduction, DerivativeWorks, databaseRight, reutilization, extraction, CommercialUse. Duties: Notice, ShareAlike, Attribution.
Process	`https://github.com/mtrebi/SentimentAnalyzer/tree/master/process/scraper.rmp`
Teams	$ILAN$, $MAPI$

(b) FOOD

FOOD	**Models for Food Rating Prediction.**
A lift chart graphically represents the improvement that a mining model provides when compared against a random guess, and measures the change in terms of a lift score. In this task, two techniques are compared, namely Decision Tree and Naive Bays. The task uses data about Food Ratings in information about quality of life in Milton Keynes Wards. The result are two pareto charts to be compared.	
Datasets	OpenDataCommunities Worthwhile 2011-2012 Average Rating (UK OGL 2.0), Food Establishments Info and Ratings (Terms of use)
Policies	Permissions: DerivativeWorks, Distribution, Reproduction, display, extraction, reutilization. Prohibitions: modify, use. Duties: Attribution, Notice, display.
Process	`https://github.com/samwar/tree/master/rapid_miner_training/16_lift_chart.rmp`
Teams	$NIFR$, $ALPA$

Table 6.1: Data Journeys (c,d).

(c) CLOUD

CLOUD	**A tag cloud from microblog posts.**
Producing statistics about keywords occurrances in text is a complex task that involves often ad-hoc cleaning operations (for instance the filtering of irrelevant textual tokens). Twitter posts about Milton Keynes are collected and processed in order to obtain a clean vector of words, associated with an occurrence score.	
Datasets	Twitter Feed #miltonkeynes (Terms of use)
Policies	Permissions: copy, display. Prohibitions: give, license, sell, transfer. Duties: attribute.
Process	`https://github.com/jccgit/RM-Textmining-Pubmed/tree/master/Pubmed.rmp`
Teams	$CAAN$, $ALPA$

(d) AVG

AVG	**Moving average of sensors' records.**
Calculation of a moving average and plotting from sensor records. In statistics, a moving average is a computation to observe data points by creating series of averages of sequential subsets of the full data set. In this task the objective is generating charts about sensor data with different time windows.	
Dataset	Samsung Sensor Data (Terms of use)
Policies	Permissions: aggregate, anonymize, archive, derive, index, read, use. Prohibitions: CommercialUse, distribute, give, grantUse, move, sell, transfer. Duties: anonymize.
Process	`https://github.com/billcary/Rapid_Miner/tree/master/chapter03/MovingAveragePlotter.rmp`
Teams	$NIFR$, $MAPI$

Table 6.1: Data Journeys (e).

(e) CLEAN

CLEAN	**Sensor data cleaning workflow.**
Cleaning sensor data workflow including devices annotated with postcodes. Input is a combination of data from all the weather stations with temperature, humidity, pressure, rainfall at different time points and space points. The process performs a number of cleaning operations on sensors streams linked with postcodes in order to obtain a dataset ready for analysis.	
Datasets	Postcode Locations (UK OGL 2.0), Netatmo Weather Station - 52.022166, -0.806386, Netatmo Weather Station - 52.002429770568, -0.79804807820062 (Terms of use)
Policies	Permissions: CommercialUse, DerivativeWorks, Distribution, Reproduction, display, extraction, reutilization, use. Prohibition: Distribution, give, grantUse, license, transfer. Duties: Attribution, Notice, inform, obtainConsent.
Process	`https://github.com/MartinSchmitzDo/RapidMinerDataCleaner/processes/clean.rmp`
Teams	$CAAN$, $ILAN$

Data analysis. We performed two different types of analysis: (a) an agreement analysis, to quantitatively measure the accuracy of the system; (b) a disagreements analysis, focused on discussing the quantitative results in the light of the users' justifications about controversial and borderline decisions, in a qualitative way. To assess the value to users of the support for policy propagation, we aggregated and discussed the responses of the questionnaire, which we present in Section 6.3. From the point of view of evaluating the accuracy and utility of the system, the quantitative data collected by the tool was expected to produce one of the following general results: a) teams agree with the system (and between each other), therefore the system is accurate; b) teams agree with each other that the system is not correct, therefore the task is feasible but the system needs to be improved; or c) users do not agree with each other, and therefore the task of supporting automatically such decision is not feasible. In this last case, in fact, we would not be able to assess the accuracy of the system, and this might be evidence that the task cannot be solved at all, or at least that the knowledge bases used by the system are not sufficient to reason on policy propagation. The accuracy analysis is reported in Section 6.4, and complemented with a discussion of its statistical significance. A qualitative analysis was conducted by focusing on the disagreements and borderline cases selected from the quantitative results. To this aim, we transcribed the notes and conversations occurred during the experiment from the audio recordings and the tool. From these data we derived

a set of general themes about fundamental issues on policy propagation, adopting a method that is akin to Grounded Theory (GT). We illustrate some exemplary cases and present the extracted themes in the discussion Section 6.5.

Table 6.2: User's feedback. The shading of the cells reflect the distribution of the answers.

Q.ID	Question						
	Left answ.	<<	<	***Unsure***	>	>>	***Right answ.***
Q.1	How difficult was it to take a single decision on whether a policy propagates to the output?						
	Easy	0	1	3	6	0	Difficult
Q.2	Do you think you had enough information to decide?						
	Yes	2	6	2	0	0	No
Q.3	How difficult was it to reach an agreement?						
	Easy	1	5	2	2	0	Difficult
Q.4	Somebody with strong technical skills is absolutely required to take this decision. Do you agree?						
	Yes	1	8	1	0	0	No
Q.5	Somebody with strong technical skills is absolutely required to take this decision even with the support of automated reasoning. Do you agree?						
	Yes	1	5	1	3		No
Q.6	Understanding the details of the process is fundamental to take a decision. Do you agree?						
	Yes	6	3	1	0	0	No
Q.7	How enjoyable was it to discuss and decide on policies and how they propagate in a process?						
	Very	4	5	0	1	0	Not
Q.8	How feasible/sustainable do you think it is to discuss and decide on policies and how they propagate in a process?						
	Feasible	3	1	3	1	2	Unfeasible
Q.9	How sustainable do you think it is to discuss and decide on policies and how they propagate in a process with the support of our system?						
	Feasible	5	4	0	1	0	Unfeasible

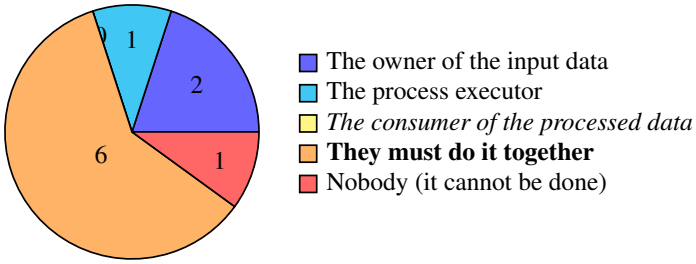


Figure 6.3: Q10. Who should decide on what policies propagate to the output of a process?

6.3 User's feedback

Before analysing the data journeys and how the decisions of the users relate to the behaviour of our system it is worth showing the feedback received after the study was conducted, collected through a questionnaire. In the questionnaire, we asked some questions about the problem of policy propagation to assess the value of the system to the user. The questionnaire was completed by the study participants individually. Table 6.2 summarises the nine closed-ended Likert-scale questions ($Q.1 - 9$), while Figure 6.3 shows the result of the single-choice question ($Q.10$). The majority of the participants in our study believe that the task can be a difficult one ($Q.1$). However, the knowledge provided was adequate for making an informed decision ($Q.2$). Deciding whether a policy propagates is possible, even if not always trivial ($Q.3$). Users agree on considering policy propagation a problem that cannot be solved without understanding the details of the data manipulation process ($Q.6$), therefore someone with strong technical skills needs to be involved ($Q.4$, $Q.5$). The objective of $Q.7$ was to check whether users were positively involved in the study, assuming that a unengaged person would not put enough effort on expressing his opinion and taking thorough decisions. Questions $Q.8$ focused on the sustainability of the task. Users feedback on this matter was spread. Our hypothesis is that two data journeys are probably not enough to understand how much this task could scale in a real setting. However, our system can effectively support the user on taking a decision ($Q.1$, $Q.9$). This feedback shows that policy propagation is a difficult problem, although it can be solved with the right knowledge models. Therefore, a tool supporting this task has good value for users. The last question ($Q.10$) was meant to understand whether the Data Hub manager could actually decide on policy propagation. It turns out that most of the users think he/she cannot solve the issue alone, but he/she should involve the data owner and the process executor in this task. This conclusion reflects some of the issues raised during the study, which is discussed in Section 6.5.

6.4 Accuracy analysis

Table 6.3: Data Journeys: System decisions

Journey	*Propagated*	*Permissions*	*Prohibitions*	*Duties*
SCRAPE	15/16	4/5	8/8	3/3
FOOD	8/22	0/12	4/4	4/6
CLOUD	5/7	0/2	4/4	1/1
AVG	8/15	0/7	7/7	1/1
CLEAN	9/17	0/8	5/5	4/4
Tot	39/77	4/34	28/28	13/15

In this section, we show how the decisions made by the users compare to the system. The decisions taken by the system are summarised in Table 6.3. For example, the $SCRAPE$ data journey required to check 16 policies and the system decided to propagate 15 of them: 4 of the 5 permissions and all the prohibitions and duties.

Tables 6.4a-6.4h summarize the results of our study in a quantitative way. The values are shown in two sets including the full numbers and the computed ratio, considering all the decisions (Tables 6.4a and 6.4b), and then split into *permissions* (Tables 6.4c and 6.4d), *prohibitions* (Tables 6.4e and 6.4f), and *duties* (Tables 6.4g and 6.4h). The values are first shown for each one of the user study (data journey of each team), aggregated for each data journey (average of both teams) and then as totals considering the decisions from all data journeys (at the bottom). The data journeys required from seven to twenty-two policies to be analysed for a total of seventy-seven *decisions*. Table 6.4a shows the number of decisions for each data journey (column D) and how much the teams agreed with the system (T_{avg} being the average value of the teams on the same data journey).

The agreement with the system is good, distributed differently across the data journeys and the teams, with an average ratio of 0.8. Moreover, this result is supported by the high agreement rate between the two teams ($T_{avg} = 0.7$). We observe that in more than half of the cases the decisions were made with the same degree of confidence (T_{12+}=0.6), and that in 70% of the cases users made a sharp decision about whether a policy would propagate or not (T_{1+}/T_{2+} total average is 0.6). Inspecting the table we see that the data journeys showing a lower agreement are $FOOD$/T_1, AVG/T_2 and $CLEAN$/T_2. We will discuss these in the next section. The low scores on $CLOUD$/T_{12+} and $CLOUD$/T_{2+} only show a difference in the degree of confidence of the decisions, which is not especially relevant in this global view, although this aspect will be discussed when looking at specific classes of policies.

Table 6.4: Agreement analysis (a,b). Tables on the left indicate totals, while the ones on the right show the related ratios.
D: total number of decisions; T_1, T_2: agreement between system and each team; T_{avg}: average agreement between teams and system; T_{12}: agreement between teams; T_{12+} agreement between teams (only *Certainly Yes/Absolutely No* answers); T_1+, T_2+: amount of *Certainly Yes/Absolutely No* answers.

(a) All decisions (totals)

Journey	D	T_1	T_2	T_{avg}	T_{12}	T_{12+}	T_{1+}	T_{2+}
SCRAPE	16	15	13	14	14	11	11	14
FOOD	22	14	18	16	14	8	20	12
CLOUD	7	5	7	6	5	1	5	2
AVG	15	15	8	11.5	8	8	15	15
CLEAN	17	12	9	10.5	14	3	13	6
All	77			58	55	31	56.5	

(b) (ratios)

T_1	T_2	T_{avg}	T_{12}	T_{12+}	T_{1+}	T_{2+}
0.9	0.8	0.9	0.9	0.8	0.7	0.9
0.6	0.8	0.7	0.6	0.6	0.9	0.5
0.7	1	0.9	0.7	0.2	0.7	0.3
1	0.5	0.8	0.5	1	1	1
0.7	0.5	0.6	0.8	0.2	0.8	0.4
		0.8	0.7	0.6	0.7	

Tables 6.4c and 6.4d only show results involving policies of type *permission*. The average agreement between the system and the users considering all the decisions is 0.6. Particularly, the $SCRAPE$ data journey for T_2 shows a low agreement (0.6), also reflected in the number of common sharp decisions (0.4). This is a low score compared with the agreement ratio of *prohibitions* (0.9) and *duties* (0.8) that can be observed in Tables 6.4f and 6.4h. It is sufficient to consider at this stage how it was much easier to take decisions on prohibitions and duties, while permissions where a greater source of discussions and disagreements with the system. Moreover, decisions about prohibitions and duties appeared to be sharper than the ones about permissions, as both the agreement between the teams (T_{12}) and the choices with strong confidence (T_{1+},T_{2+}) received higher scores. However, on both types of policies, the source of disagreement is on the $FOOD$ data journey. We showed that this is the case ($\tilde{8}0\%$ agreement). We complement this data with a statistical analysis based on the Cohen's kappa coefficient (CKC), which takes into account the possibility of the agreement occurring by chance. The 95% Confidence Interval (CI) of CKC between the system and either human teams T1 or T2, is not significantly different from the 95% CI of CKC between T1 and T2. In other words, the system behaves as a user would do, also from a statistical point of view.

Table 6.4: Agreement analysis (c,d,e,f,g,h). Tables on the left indicate totals, while the ones on the right show the related ratios.
D: total number of decisions; T_1, T_2: agreement between system and each team; T_{avg}: average agreement between teams and system; T_{12}: agreement between teams; T_{12+} agreement between teams (only *Certainly Yes/Absolutely No* answers); T_1+, T_2+: amount of *Certainly Yes/Absolutely No* answers.

(c) Permissions (totals)

Journey	D	T_1	T_2	T_{avg}	T_{12}	T_{12+}	T_{1+}	T_{2+}
SCRAPE	5	4	2	3	3	0	0	3
FOOD	12	12	12	12	12	6	12	6
CLOUD	2	0	2	1	0	0	0	1
AVG	7	7	0	3.5	0	0	7	7
CLEAN	8	3	0	1.5	5	2	4	5
All	34			21	20	8	22.5	

(d) (ratios)

T_1	T_2	T_{avg}	T_{12}	T_{12+}	T_{1+}	T_{2+}
0.8	0.4	0.6	0.6	0	0	0.6
1	1	1	1	0.5	1	0.5
0	1	0.5	0	N.A.	0	0.5
1	0	0.5	0	N.A.	1	1
0.4	0	0.2	0.6	0.4	0.5	0.6
		0.6	0.6	0.4	0.7	

(e) Prohibitions (totals)

Journey	D	T_1	T_2	T_{avg}	T_{12}	T_{12+}	T_{1+}	T_{2+}
SCRAPE	8	8	8	8	8	8	8	8
FOOD	4	0	2	1	2	2	4	2
CLOUD	4	4	4	4	4	0	4	0
AVG	7	7	7	7	7	7	7	7
CLEAN	5	5	5	5	5	0	5	0
All	28			25	26	17	22.5	

(f) (ratios)

T_1	T_2	T_{avg}	T_{12}	T_{12+}	T_{1+}	T_{2+}
1	1	1	1	1	1	1
0	0.5	0.3	0.5	1	1	0.5
1	1	1	1	0	1	0
1	1	1	1	1	1	1
1	1	1	1	0	1	0
		0.9	0.9	0.7	0.8	

(g) Duties (totals)

Journey	D	T_1	T_2	T_{avg}	T_{12}	T_{12+}	T_{1+}	T_{2+}
SCRAPE	3	3	3	3	3	3	3	3
FOOD	6	2	4	3	0	0	4	4
CLOUD	1	1	1	1	1	1	1	1
AVG	1	1	1	1	1	1	1	1
CLEAN	4	4	4	4	4	1	4	1
All	15			12	9	6	11.5	

(h) (ratios)

T_1	T_2	T_{avg}	T_{12}	T_{12+}	T_{1+}	T_{2+}
1	1	1	1	1	1	1
0.3	0.7	0.5	0	N.A.	0.7	0.7
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	0.3	1	0.3
		0.8	0.6	0.7	0.8	

6.5 Discussion

The results show that the task is feasible in all the scenarios and that our system exhibits good accuracy. In what follows we analyse the cases in which users disagreed with the system (also highlighted in Tables 6.4a-6.4h).

We expected three types of disagreements: a) the system is missing a policy; b) the system should block a policy; c) the system should not decide about it as it does not have enough information. We note that option (c) never emerged from the study. The teams always made a clear decision whether to propagate a policy or not[13].

The $SCRAPE$ data journey. Both teams agreed that the permission to `lds:extract` must be propagated. The system explanation showed that the policy was actually blocked by `dn:hasExtraction`. Both teams indicated this as an error, identifying the anomaly that was intentionally introduced, reassuring us about the commitment in performing the task. $MAPI/T_2$ disagreed about propagating two specific permissions: `cc:Distribution` and `ldr:reutilization`. Although the general activity was one of website crawling and indexing, $MAPI/T_2$ considered the type of indexing implemented to affect the interpretation of the content of the website, in such a way to potentially damage the interests of the original owner: *"The process has a step in which some values were changed and then these changed values are assigned to be the LABELS of the items. [...] The permission to distribute of the new output should not be given for granted, to protect the owner of the content. This choice changes depending on the content of the data and not on the general action performed"*. What is relevant here is not the point itself, which is debatable, but the fact that the participants believed the process and the policies were not enough to decide whether to propagate the policy.

The $FOOD$ data journey. This process produced two outputs, a *performance dataset*, including performance vectors of the machine learning algorithms compared, and a consequent *lift chart*, i.e. a graphical representation of the data. The system did not propagate any of the permissions (this decision aligns with the two teams). However, $NIFR/T_1$ changed their decision *after* seeing the explanation given by the system. Moreover, $NIFR/T_1$ decided that no policy should propagate to the output of the process, while $ALPA/T_2$ agreed with the system that both prohibitions and duties must be preserved for the performance dataset output and only the duties in the case of the *lift chart*. This difference is motivated by the interpretation of the nature of the *performance dataset*. By analysing the conversation occurred in the user study, it emerges that $NIFR/T_1$ considered the dataset containing only measures of the performance of the algorithms, while $ALPA/T_2$ interpreted

[13] Although one participant observed in the questionnaire that the task of deciding on the policies to apply to a derived dataset was impossible.

it as a labelled dataset, therefore containing an enhanced version of the input data. The correct interpretation is the one of $NIFR/T_1$, once this is reflected in the data flow, the system will block all the policies as they are not applicable to the *performance dataset*.

The $CLOUD$ data journey. $CAAN/T_1$ disagreed on the behaviour of a set of rules about two permissions: `odrl:copy` and `odrl:display`, and marked as wrong the behaviour of: `dn:processedInto`, `dn:cleanedInto`, `dn:refactoredInto`, `dn:isPortionOf`, and `dn:combinedIn`. In particular: *"combinedIn should propagate because both of the inputs have the permission to copy, in case one of the two has not, it shouldn't. You need to reason on the combination to decide the propagation."* During the session, the team proposed to propagate the permissions to the combined node as soon as no prohibition is present.

The AVG data journey. $NIFR/T_1$ changed their mind about permissions after seeing the explanation of the reasoner. Since the dataset was *modified* it made sense that the permissions were not propagated. However, $MAPI/T_2$ disagreed with both the system and the other team, and identified the problem by inspecting the explanation of the system: relation `dn:remodelledTo` must propagate the various permissions: *"As far as we understood there is no bias introduced in the data, therefore the permission should be kept intact. The outcome of the process depends entirely on the input, without additional information. It's just a mathematical process that keeps the information intact"*. In fact, `dn:remodelledTo` is defined as "Remodelling refer to the translation of the data to another symbolic structure (model), while keeping the same interpretation". This is a case where the teams disagreed about the system. However, the justification of $MAPI/T_2$ seems robust enough to accept the change of behavior of the `dn:remodelledTo` relation.

The $CLEAN$ data journey. Both teams observed that `dn:combinedIn` should propagate the permissions involved while the system decided not to. The main argument was that the relation should consider the policies of all datasets involved in the combination, however, without knowing them, the system should propagate them and leave the decision to a consistency check to be applied at the later stage. In another discussion, it was observed how in some cases there is a dependency between policies. It is the case of *duties*, which are always in the context of a permission, therefore by propagating the former, the system should also propagate the latter. For example, the permission to use should be propagated as a dependency to the duty to obtain consent.

The issues illustrated can be grouped under the following general themes:
a) *Incomplete knowledge.* The knowledge base used by the system is not complete: rules should be added or modified in order to *fix* the behaviour with respect to certain policies and relations, using the methodology presented in Chapter 4. Data flows should be accurate and include all the relevant relations.

b) *Data reverse engineering.* A recurrent theme for assessing whether permissions should propagate was the contingency of *data reverse engineering*, defined in software engineering as "the use of structured techniques to reconstitute the data assets of an existing system" [Aiken (1996)]. We observe that the correct interpretation of the nature of the output is crucial. Therefore, it is of fundamental importance that the data flow description is accurate, including assessing how much the information of the input data source can be extracted from the output data. In some cases, the implemented data flow was not complete enough to reflect this issue.
c) *Content-dependent decisions.* It was argued that in one case the impact of the process on the output policies could not be assessed without inspecting the content of the data. We cannot argue against this in principle. However, we assume that new relations could be developed within the methodology of [Daga et al. (2015a)] in order to capture fine-grained implications of process actions on policy propagation, making this a case of incomplete knowledge base.
d) *Dependant policies.* The approach of the system was to consider the policies in isolation, and focusing on their interaction with process actions. However, it is clear that policies on their own incorporate a number of dependencies, some of them derived from the semantics of the action involved (for example `odrl:copy` is a kind of `odrl:use`), others from the way they are formalised in policy documents (in ODRL, a duty is always declared in the context of a permission). See also [Steyskal and Polleres (2015)] for a discussion on this. However, by knowing that a policy needs to be taken into account on the output of a given process, dependant policies can be extracted from the original policy document.
e) *The Legal Knowledge.* A general observation that many of the participants made is that this is a *legal* issue, therefore a legal expert should be involved in the definition of policy actions, process descriptors, and PPRs. On one hand, this suggests the importance of providing support to Data Hub managers on deciding on policy propagation, as we cannot expect this type of users to have legal knowledge. On the other hand, this highlights a more general issue. In fact, a validation of the system by legal experts would assume a legal framework covering the status of metadata-oriented automatic reasoners in the Rule of Law, which is currently missing [Casanovas (2015)].

6.6 Conclusions

In this work, we evaluated an approach and a system to support the assessment of the policies propagating from a data source to a derived dataset in a Data Hub. Participants agreed that it is possible to decide whether a policy associated with a dataset needs to be associated with another derived dataset. The results of our user study demonstrated that the task can be solved automatically

with a good degree of accuracy. By considering both the results of the user study and the feedback collected, the system is overall accurate and is of good value to users. The study also let emerge a set of critical aspects involved. It is important that the knowledge bases are complete, in particular, that the process description does not hide any of the elements that could influence the propagation of policies, for example making it clear how much of the data of the input can be extracted from the output.

However, more research is required in order to include in the knowledge base other aspects involved in policy reasoning. For example, the rights of other stakeholders should be involved in the process, including the ones of the process executor (what action adds value to the information?), or the rights of the entities represented in the data (from businesses to private citizens).

Chapter 7

Conclusions and Outlook

The objective of this work was to show how it becomes possible to reason about the propagation of policies in data flows once we are capable of developing the appropriate knowledge components. Initially, we focused on how these components can be defined, acquired, and used to reason on the propagation of policies (Chapters 3-4). We then studied how policy propagation can be part of the *life-cycle* of data objects, from initial publication to their use in processed forms. In particular, we proposed a methodology and a toolkit for supporting the users involved in the various tasks associated with the management of policy and process metadata (Chapter 5). Finally, we performed a user study aimed at evaluating our approach, showing that deciding on the propagation of policies can indeed be achieved by developing and curating the required *knowledge components* (Chapter 6). By doing this, we answered the following questions:

- $R.Q.\ 1$ What are the knowledge components required to reason on policy propagation?
- $R.Q.\ 2$ How should we represent the relation between processes and policies in order to practically support the task of reasoning about policy propagation?
- $R.Q.\ 3$ How can we assist users in setting up the necessary components to enable automated reasoning on policy propagation?

Our approach was based on a set of hypotheses:

- $H.\ 1$ It is possible to design a system capable of propagating policies in data flows by following a Knowledge Engineering approach..
- $H.\ 2$ An abstract representation of a data manipulation process is sufficient for reasoning on policy propagation. Such abstraction characterises the actions performed as relations between objects in a RDF graph.
- $H.\ 3$ This type of reasoning can be done combining rules, transitivity, and RDFS inferences.

– $H.\ 4$ A semantic model of processes and policies, while being necessary to reason about policy propagation, can also support the acquisition of policy and process knowledge.

In the next section, we provide an overview of our contribution to addressing the research questions and discuss to what extent our initial hypotheses have been verified. The proposed solutions have been developed under a set of assumptions, therefore we review them and discuss the limitations of our approach in Section 7.2. While doing this, we also discuss the opportunities that our study enables and the major challenges that future research will need to address.

7.1 Overview of Contributions and Major Findings

Our contribution focuses on the *knowledge components* required to reason upon the propagation of policies in data flows, therefore the initial research question was centred on identifying them. In particular, the fundamental hypothesis was that *it is possible to design a system capable of propagating policies in data flows by following a Knowledge Engineering approach* ($H.\ 1$). This introduces our first research question:

$R.Q.\ 1$: What are the knowledge components required to reason on policy propagation?

In the introduction we listed these components in terms of what they are supposed to require in support of the task, and we expressed them in the following way: *(a) information about the data sources, ownership and licence; (b) information about the process manipulating the data sources; (c) information about how process steps affect policies at a granular level.*

The first component is a formalised description of the data sources involved and their licences. We surveyed the state of the art and reported on how data and the associated metadata can be represented, but our contribution did not focus on this aspect. Similarly, we did not contribute on the topic of licence representation, and relied on the assumption that ODRL as a language is sufficient to represent policy statements derived from licences, as argued in the literature (e.g. [Cardellino et al. (2014); Governatori et al. (2014); Rodríguez-Doncel et al. (2014)]). In our work, we relied on DCAT as the reference specification for cataloguing datasets, and on ODRL to represent licences, in particular developing on top of the RDF Licences Database [Rodríguez-Doncel et al. (2014)].

However, a representation of the licence metadata cannot be sufficient to reason on how policies propagate to other data artefacts. We hypothesised that *an abstract representation of a data manipulation process is sufficient for reasoning on policy propagation* ($H.\ 2$). Therefore, the second component is dedicated to the representation of the *processes* in the Web of Data. We dedicated a significant part of our literature review to surveying the various approaches related to process

modelling. We observed that the centre of them was the concept of *operator*, a "black box" that consumes and produces data, connected to other operators, composing rich data pipelines. However, we encountered two problems: 1) these operators can ingest and deliver many data objects, and therefore we cannot easily derive what are the dependencies between the various data items involved; 2) the processes we care about are the ones happening on the Web of Data, but how to elicit them? We needed a way to access a type of knowledge that is by definition managed in a distributed fashion, where data providers, processors and consumers are not directly available.

To tackle the first problem, we conjectured that *such abstraction could characterise the actions performed as relations between objects in a RDF graph* ($H.\ 2$). In Chapter 3 we propose a data-centric approach to the representation of data flows, to complement operator-centric representations. The Datanode ontology allows us to represent processes as networks of data objects, therefore making it possible to analyse the dependencies between them, and characterise the semantics derivable from data manipulation steps. The types of possible relations were acquired by selecting a number of Semantic Web intelligent systems. Because we could not access the Web of Data per se, we used those systems as exemplary prototypes of Web of Data processes, addressing the second problem mentioned before.

With Datanode we can represent the relations between the data objects. However, this is not enough for reasoning on policy propagation. Therefore, we made the hypothesis that this type of reasoning can be done combining rules, transitivity, and RDFS inferences ($H.\ 3$). To validate this hypothesis, at the beginning of Chapter 4, we established Policy Propagation Rules (PPRs) as a means to specify whether a certain relation among two data objects *triggers* the propagation of a certain policy. We were capable of selecting a number of realistic policies from the RDF Licenses Database and associated them with Datanode relations. The set of PPRs are therefore our third *knowledge component*, specifying how process steps affect policies at a granular level. However, this leaves open a second important question:

$R.Q.\ 2$: How to represent the relation between processes and policies in order to *practically* support the task of reasoning on policy propagation?

At this stage of the work, we identified and developed a set of components, but are they suitable for policy propagation? Our research question puts an emphasis on the *practicality* of the approach supporting the task of policy propagation. This has been approached through several angles in Chapter 4:

– By applying our approach, a large number of PPRs can be produced. We propose the (A)AAAA methodology, which exploits FCA in combination with the Datanode ontology to reduce the number of rules to be managed.

– The Datanode ontology has been designed to represent the way data objects are manipulated in the Web of Data, but we did not know whether its property hierarchy was consistent with the semantics of the relations and their behaviour in the PPRs rule base. In Chapter 4 we evolved Datanode and validated it with respect to the PPRs. We can use them together to reason upon policy propagation.
– To demonstrate that this reasoning is feasible, we developed a PPR reasoner and reasoned on the propagation of policies by relying on a set of Semantic Web application as reference scenarios. Our reasoner combines production rules, transitivity, and RDFS entailments.
– We also investigated whether rule compression affects the performance of a PPR reasoner. We performed a set of experiments by relying on two different approaches to reasoning, (a) runtime rule execution (implemented in Prolog) and (b) load time inference materialisation (using OWL reasoning and SPIN rules). These experiments demonstrated that not only performance is not affected negatively, but also that in both cases there is an improvement in efficiency.

So far, we demonstrated that reasoning upon the propagation of policies is possible with transitivity and RDFS inferences, once the appropriate knowledge components are developed. This validated the hypotheses $H.\ 1$, $H.\ 2$, and $H.\ 3$. At this stage, we had a tool to reason on policy propagation, but we did not know whether the approach behind it was sustainable in realistic scenarios. We were capable of building such components, but is our approach portable to other users? This brings us to the last research question:

$R.Q.\ 3$: How can we assist the users in setting up the necessary components to enable automated reasoning on policy propagation?

To answer this question, we formulated the hypothesis that *a semantic model of processes and policies, while being necessary to reason about policy propagation, can also support the acquisition of policy and process knowledge* ($H.\ 4$). In Chapter 5, we took a Smart City Data Hub as a reference scenario for answering this question and developing our solutions. The MK Smart Data Hub is a Web of Data *in vitro*, where data publishers and data consumers collaborate by means of a mediating processing infrastructure, curated and maintained by a data manager. Such an environment gave us the opportunity to identify two major bottlenecks in the acquisition of the required knowledge. First of all, users need to associate terms and conditions to the data they intend to publish, among the ones available. Once again, we do not focus on the acquisition of licence formalisation, assume a catalogue of ODRL descriptions exists, and use the one provided by the RDF Licenses Database. Section 5.2 addressed this problem by exploiting a semantic representation

of licences in combination with FCA, and reduced the burden of licence selection to answer a small set of intuitive questions. The second problem was the one of producing a description of the operations performed by the Data Hub when generating derived datasets. Generating Datanode descriptions is a time-consuming activity even considering the existence of standardised process models, like the one developed for the execution and reuse of Scientific Workflows. In Section 5.3, we proposed to support users in generating Datanode descriptions by annotating existing processes, taking Myexperiments.org as an exemplary source, and embedding the characteristics of typical process models. The approach we take is the one of a recommendation system, developed by incrementally mining association rules with the support of FCA. We demonstrated that this can be done without the existence of a reference gold standard of Datanode descriptions. In particular, we proved that a semantic representation of the process (in particular its *metadata*), can be leveraged to support users in the acquisition of process knowledge. In the end, both policy and process models acquisition can be supported by relying on their semantic representation, as envisaged in our Hypothesis $H.\,4$.

However, although we were capable of supporting users in the acquisition of the components, we still missed the whole picture about their *setup* in a realistic scenario. The MK Data Hub was also the setting where the solutions for policy propagations developed in Chapter 4 could be validated, in the light of a comprehensive methodology for policy propagation. Such methodology, which we called Metadata Supply Chain Management (MSCM), establishes a parallel workflow to the one involving the actual data objects, which pertains their related *metadata*, and particularly the policies associated with them. By validating our *knowledge components* in the context of a real Smart City Data Hub, we were able to characterise them better. At the end of our journey, we integrated our components in a unified *Data Catalogue* under a *holistic* perspective:

a) a Licences Catalogue, providing a semantic representation of licences as a set of atomic ODRL policies;
b) a Dataflows Catalogue, providing a semantic representation of the relations between data objects;
c) a Policy Propagation Rules database, providing a collection of rules that enable policies to propagate among data-to-data relations.

These knowledge components have been identified, developed and practically tested in a realistic scenario. However, we had not evaluated yet to what extent policy propagation assessment as a task can be reduced to reasoning over binary associations between the data objects involved. In other words, would the results of such reasoning be meaningful for the user who typically perform such a task? In particular, would the decisions of a data manager with regards to the policies applicable to

a derived dataset be coherent with our system? The methodology described in Section 5.4 formed the basis for which we designed a user study aimed at two major objectives:

1. to perform an empirical evaluation of the tool support for policy propagation, following the guidelines developed in our MSCM methodology, measured in terms of its *accuracy* and perceived *user value*.
2. to explore the feasibility of the task of deciding on policy propagation, particularly regarding the sufficiency of the knowledge components we identified.

For this purpose, we developed the user study reported in Chapter 6, where we showed that the developed system is accurate and of good value to users. This result encourages us to think of policy propagation as a special type of *metadata* propagation between interlinked datasets. Moreover, by engaging with users whose knowledge and skills are similar to the ones of users who typically would perform the task of deciding on policy propagation, we were able to explore some borderline cases that could make emerge some limitations of our approach. Among others, these are discussed in the next Section.

7.2 Limitations of the Study and Future Directions

We demonstrated that the problem of reasoning upon policy propagation is a tractable one. However, there are issues coming from the limits of our approach that deserve to be discussed, as they enable new opportunities for further research.

About the completeness of Datanode. Datanode has been designed taking several systems and studying their features. However, we cannot demonstrate that the hierarchy of possible relations between data items is complete. This is a common problem in ontology development, and we strongly believe that it can only be tackled from the perspective of *quality* and *evolution*. The *quality* of the ontology can be considered as *fitness for purpose*. In the case of Datanode, this has been assessed both technically (Chapter 4) and methodologically (Chapter 5-6). However, relations might emerge from the analysis of processes, which could lead to the *evolution* of the hierarchy. The (A)AAAA methodology would need to be adjusted in order to support the continuous integration of ontology changes in its workflow. Although we haven't specifically studied this aspect, the methodology already contains an iterative workflow, and have elements that can be reused for that purpose, such as the coherency check between the ontology and the PPRs.

Binary relations are not always sufficient. We developed Datanode under the hypothesis that systems can be described as networks of data objects, and that this type of model would have been sufficient to reason upon policy propagation. Although we demonstrated that there is a correspondence between the types of data-to-data relations and policy propagation, this knowledge is not *always* sufficient. One example is the behaviour of `dn:combinedIn` in relation to *permissions*, as discussed in Section 6.5. In fact, `dn:combinedIn` implies that a set of data items are composed into a new one. It was argued by the user study participants that a given *permission* policy must propagate only if it was shared by all the involved data objects. Therefore, the `dn:combinedIn` relation should not be binary. One possible solution to this problem would be to extend the rule base to support more sophisticated rules than the binary predicate *propagates*. Another one is to apply other inference methods, for example adopting *probabilistic logic*, and trigger the propagation by assigning a degree of uncertainty.

About licence decomposition. We developed our solution under the assumption that licences can be actually reduced to sets of atomic policies. However, there are cases in which the policies need to be reasoned upon as combined sets. For example, propagating the duty to attribute does not make sense outside the scope of a permission (and is also inconsistent with version 2.1 of the ODRL specification [Iannella et al. (2015)]). The problem could be addressed by structuring the license as a graph of dependencies between atomic policies. However, it is unclear how these dependencies would affect the propagation. Intuitively, propagating a permission would probably carry a dependent duty, but not vice versa. However, the interaction between composite policies and the process might make other issues emerge.

Reasoning over the compatibility of propagated policies. In the present work, we do not cover the analysis of the compatibility between the propagated policies. Although in several occasions we stressed the fact that policy propagation is the necessary preliminary step for assessing the policies to apply to the output of a process, it can be argued that a real decision cannot be made without also considering their compatibility. In fact, this step is considered in the tool chain and included in the methodology proposed in Section 5.4. However, it was left out of the user study to avoid participants being distracted by discussing what could have been the final policy set to apply to the data, instead of focusing on the actual task of verifying what policies would have been necessary to propagate.

Process executions can influence policy propagation. We developed our solution under the assumption that policy propagation can be reasoned at *design time* and that process executions

would not affect policy propagation. Our general scenario assumes the process description to be designed once and for all, and to be valid for any possible data input (see Table 5.5). Although knowing the content of a dataset is not necessary to apply our reasoner, there might be aspects of the Datanode descriptions that could depend on the input data. In fact, workflows could perform alternative operations depending on the *content* of the data. In these cases, policy propagation cannot be reasoned completely on simulated executions but would need to be performed on Datanode descriptions of execution *traces*. Therefore, an improved version of the MSCM methodology should also consider execution traces, and support the user in two stages: a) at design time, to help her/him to assess what policies should propagate using a generic Datanode representation; and b) at runtime, to evaluate the propagation with respect to the actual Datanode execution trace. Moreover, an incremental log of execution traces could be used to generate a more accurate generic Datanode representation, by considering the history of the implications of runtime executions of the process on different data policies.

About the validation of processes. In our study, we did not consider whether the processes are valid with respect to the data licences involved. In other words, we assume the data consumer of the derived data to be the party to which the licence is directed. By doing that, we assumed that the data processor always has the right to execute the process (see also Table 5.5). An interesting future work would be to consider the validation of the *process*. There are a number of research questions that deserve to be inspected, for example:

- How can we diagnose inconsistencies between a data flow and the related data policies?
- How to support the user in selecting the processes that are compatible with the policies of the input data sources and the legal requirement of the user's task?

About the accountability of licence descriptions. It can be argued that a Licence Catalogue should be *authoritative* so that the coherency of the ODRL descriptions with the actual licence documents could be trusted. Although part of the problem is non-technical by nature, and pertains the legal status of formalised knowledge in general, an important contribution could be made by studying methods and tools to support the life cycle of these legal models, including the assessment of their trustworthiness, for example supporting their negotiation in the development of contractual agreements by the parties. However, these aspects are complementary to the problem of policy propagation.

About description logics. We approached the problem of policy propagation from a pragmatic perspective and proposed a solution developed with a design and test methodology (see Section 1.5).

Our contribution leaves space for other strands of research on how processes and policies may interact from a theoretical perspective. Some of the limitations of our approach could be resolved by studying the possible types of logics that can be adopted to reason on policy propagation, for example combining *deontic* and *probabilistic* description logics [Toni (2008)], or integrating policy propagation reasoning in a larger framework for legal argumentation [Feteris (2017)]. Overall, we have not covered the aspects of integrating the proposed knowledge components with models developed for the legal domain, for example discussing how propagation rules could be shared or negotiated as *arguments* adopting the Legal Knowledge Interchange Format (LKIF) [Gordon (2008)].

The legal value of semantic models. In our work, we purposely avoided confronting with epistemological aspects involving the *legal value* of an automated system that reasons with policies and processes. Although ICT systems play a fundamental role in the life cycle of legal knowledge, from the legislative process to the variety of applications concerning the discovery and consumption of legal documents [Kiškis and Petrauskas (2004)], computers seem to still be confined to the role of document managers, rather than *knowledge interpreters*. Despite the extensive work done in the past on understanding how foundational aspects of the law can be represented in a computable fashion [Casanovas et al. (2011, 2008); Griffo et al. (2015)], there is still a gap between academic advances and legal practice [Casanovas et al. (2016a)]. One reason could lay in the absence of an agreed theory about the status of formal models within legal argumentation. Such a theory would necessarily cover an epistemic model of coherence [Amaya (2007)], and satisfy claims such the one that coherence itself might inherently resist formalisation [Wintgens (2005)]. The problem of the legal status of AI systems is currently a hot topic, and we can hope that further research would help in filling the gap between the technical advances and the societal competence.

Semantic Web Applications, City Data Hubs, Scientific Workflows are *not* the Web of Data. Although these three scenarios include elements that are constituent of the Web of Data, there are still important differences. In particular, we developed our research under the assumption that these components can be integrated into the life cycle of data ($A.\ 2$). For example, we assumed that the infrastructure processing the data was the same as the one managing it (for example see the Acquisition phase of our MSCM methodology, Table 5.4). Although we performed a validation in the context of a Smart City Data Hub, more challenges need to be addressed in order for the approach to scale up to the Web of Data, and this is the area where we think future research on policy propagation needs to focus. In addition, data publishers, data consumers, and data managers are only (useful) abstractions. In the Web of Data, the qualities of relations between the several

actors can vary and can affect how the policies are negotiated, assigned, revoked and so forth. This has been particularly stressed in the *Onboarding* phase of the MSCM methodology (see Table 5.3), where it was assumed that a data provider granted the same licence to whoever exploits the given data source. Although this seems to be valid for open datasets, we acknowledge the fact that this can be a limiting feature for business cases in which APIs and datasets are accessible only to selected consumers. Future work includes the support of multiple licenses, for example by enabling "scopes" of use as additional metadata, and user profiling in order to add more contextual information to the reasoning process. Also, in our experiments, we assume that the data manager has the right to decide on policy propagation, while we have seen that the majority of the users involved in the study reported in Chapter 6 believe that this decision cannot be taken by a single role, but needs to be *negotiated* by multiple parties.

In conclusion, governing the life-cycle of policy metadata on the Web is multi-faceted and open to further investigation. The main issue concerns, in fact, the way in which metadata cataloguing could be developed distributedly. The knowledge components proposed here could be managed by several decentralised *mediators*. Such mediators should be capable of supporting users in the negotiation of contractual agreements on the use of information through technologies such as distributed ledgers for registering transaction traces [English et al. (2016)]. Therefore, under the perspective of social machines [Hendler and Berners-Lee (2010)], unifying both computational and social processes, a new generation of Web systems would control (meta)data in terms of *supply* and *demand*, according to the needs of the various actors involved, within an open interlinked *market* of data and processes.

Appendix A

Summary of developed software and ontologies

In this appendix we summarise the software and ontologies used and produced in the work. Ontologies are published following the Linked Data principles. References and DOIs are listed in Table A.1.

We developed several scripts to support the methodology described in Chapter 3. The software is collected in the Datanode Making project and be found at this persistent URL: `http://purl.org/datanode/tools/datanode-making`. The text snippets used and related Datanode representations are included. The Datanode ontology resulting from the methodology of Chapter 3 can be found at this persistent URL: `http://purl.org/datanode/0.3/ns/`. The documentation also includes the examples referenced in the text: `http://purl.org/datanode/0.3/docs`. The command line tool developed to support the (A)AAAA methodology (Chapter 4) can be found at this URL: `http://purl.org/datanode/tools/ppr-a-five`. The project includes the datanode ontology and an account of all the changes performed with this command line tool. The Datanode ontology resulting form the analysis performed with the (A)AAAA methodology (Chapter 4) can be found at this persistent URL: `http://purl.org/datanode/0.5/ns/`. The PP Reasoner used in the experiments of Chapter 4 can be found at this URL: `http://purl.org/datanode/tools/ppreasoner`. The project includes both the Prolog and the SPIN reasoners and can be directly reused in third-parties applications. The work in Chapter 5 is focused on supporting users on developing the required knowledge components. Contento, a tool to support the acquisition of formal contexts and the exploration of FCA lattices. The software can be found at this URL: `http://purl.org/datanode/tools/contento`. The Licence Selection tool can be found at this URL: `http://purl.org/datanode/tools/licence-picker`. The

Table A.1: List of software and ontologies

Name	Ch.	pURL	DOI	Licence
Datanode Making	Ch. 3	http://purl.org/datanode/tools/datanode-making	10.5281/zenodo.1293967	CC-A-4.0
Datanode v0.3	Ch. 3	http://purl.org/datanode/0.3/ns/	10.5281/zenodo.1293989	CC-A-4.0
Datanode v0.3 (Doc)	Ch. 3	http://purl.org/datanode/0.3/docs	-	-
PPR-A-5	Ch. 4	http://purl.org/datanode/tools/ppr-a-five	10.5281/zenodo.1293993	CC-A-4.0
Datanode v0.5	Ch. 4	http://purl.org/datanode/0.5/ns/	10.5281/zenodo.1293998	CC-A-4.0
Contento	Ch. 5	http://purl.org/datanode/tools/contento	10.5281/zenodo.1294004	CC-A-4.0
Licence Picker	Ch. 5	http://purl.org/datanode/tools/licence-picker	10.5281/zenodo.1294010	CC-A-4.0
LiPiO	Ch.5	http://purl.org/datanode/lipio	10.5281/zenodo.1294018	CC-A-4.0
Dinowolf	Ch. 5	http://purl.org/datanode/tools/dinowolf	10.5281/zenodo.1294020	CC-A-4.0
User study	Ch. 6	http://purl.org/datanode/tools/ppr-datahub-study	10.5281/zenodo.1294023	CC-A-4.0

Licence Picker ontology can be downloaded at: `http://purl.org/datanode/lipio`. The Dinowolf software developed for evaluating the incremental learning approach to association rules generation can be found at this URL: `http://purl.org/datanode/tools/dinowolf`. Finally, we developed a ad-hoc tool for performing the user study described in Chapter 6. It is published at this URL: `http://purl.org/datanode/tools/ppr-datahub-study`.

Bibliography

Ackerman, W. B. (1982). Data flow languages. *Computer*, 2(15):15–25.

Adamou, A. and d'Aquin, M. (2015). On requirements for federated data integration as a compilation process. In [Berendt et al. (2015)], pages 75–80.

Aiken, P. H. (1996). *Data reverse engineering: slaying the legacy dragon*. McGraw-Hill Companies.

Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., and Verma, K. (2005). Web service semantics - WSDL-S. Technical Report 4-2005, The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis).

Alam, M. and Napoli, A. (2015). Navigating and exploring rdf data using formal concept analysis. In *Proceedings International Workshop" Linked Data for Knowledge Discovery colocated with the ECML/PKDD*.

Alani, H., Kim, S., Millard, D. E., Weal, M. J., Hall, W., Lewis, P. H., and Shadbolt, N. R. (2003). Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems*, 18(1):14–21.

Alexander, B. (2009). LKIF core: Principled ontology development for the legal domain. *Law, ontologies and the semantic web: channelling the legal information flood*, 188:21.

Alexander, K. and Hausenblas, M. (2009). Describing linked datasets-on the design and usage of void, the'vocabulary of interlinked datasets. In *In Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09*. Citeseer.

Allocca, C., d'Aquin, M., and Motta, E. (2009). DOOR - towards a formalization of ontology relations. In *Proc. of the First Conference on Knowledge Engineering and Ontology Development*, pages 13–20. INSTICC Press.

Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2004). Web services. In *Web Services*, pages 123–149. Springer.

Alper, P., Belhajjame, K., Goble, C. A., and Karagoz, P. (2014). Labelflow: Exploiting workflow provenance to surface scientific data provenance. In *International Provenance and Annotation Workshop*, pages 84–96. Springer.

Alspaugh, T. A., Asuncion, H. U., and Scacchi, W. (2009). Analyzing software licenses in open architecture software systems. In *Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, pages 54–57. IEEE Computer Society.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., and Mock, S. (2004). Kepler: an extensible system for design and execution of scientific workflows. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 423–424. IEEE.

Amaya, A. (2007). Formal models of coherence and legal epistemology. *Artificial intelligence and law*, 15(4):429–447.

Amorim, R. C., Castro, J. A., da Silva, J. R., and Ribeiro, C. (2016). A comparison of research data management platforms: architecture, flexible metadata and interoperability. *Universal Access in the Information Society*, pages 1–12.

Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D. L., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., et al. (2001). DAML-S: Semantic markup for web services. In *Proceedings of the First International Conference on Semantic Web Working*, pages 411–430. CEUR-WS. org.

Antoniou, G. and Van Harmelen, F. (2004). Web Ontology Language: OWL. In *Handbook on ontologies*, pages 67–92. Springer.

Arndt, R., Troncy, R., Staab, S., Hardman, L., and Vacura, M. (2007). *COMM: designing a well-founded multimedia ontology for the web*. Springer.

Assaf, A., Troncy, R., and Senart, A. (2015). HDL - towards a harmonized dataset model for open data portals. In [Berendt et al. (2015)], pages 62–74.

Astrakhantsev, N. and Turdakov, D. Y. (2013). Automatic construction and enrichment of informal ontologies: A survey. *Programming and computer software*, 39(1):34–42.

Atemezing, G. A., Villazón-Terrazas, B., and Hyland, B. (2014). Best practices for publishing linked data. W3C note, W3C. http://www.w3.org/TR/2014/NOTE-ld-bp-20140109/.

Attard, J., Orlandi, F., Scerri, S., and Auer, S. (2015). A systematic review of open government data initiatives. *Government Information Quarterly*, 32(4):399–418.

Auer, S. and Hellmann, S. (2012). The Web of data: Decentralized, collaborative, interlinked and interoperable. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC-2012)*.

Aussenac-Gilles, N. and Gandon, F. (2013). From the knowledge acquisition bottleneck to the knowledge acquisition overflow: A brief French history of knowledge acquisition. *International Journal of Human-Computer Studies*, 71(2):157–165.

Ball, A., Ashley, K., McCann, P., Molloy, L., and Van den Eynden, V. (2014). Show me the data: The pilot uk research data registry. In *9th International Digital Curation Conference*. University of Bath.

Barrett, D. J. (2008). *MediaWiki*. " O'Reilly Media, Inc.".

Battle, R. and Kolas, D. (2012). Enabling the geospatial semantic web with parliament and geosparql. *Semantic Web*, 3(4):355–370.

Beagle, D. R., Bailey, D. R., and Tierney, B. (2006). *Information commons handbook*. ALA Neal-Schuman.

Belhajjame, K., Zhao, J., Garijo, D., Garrido, A., Soiland-Reyes, S., Alper, P., and Corcho, O. (2013). A workflow PROV-corpus based on taverna and wings. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 331–332. ACM.

Bellini, P., Bertocci, L., Betti, F., and Nesi, P. (2016). Rights enforcement and licensing understanding for RDF stores aggregating open and private data sets. In *Smart Cities Conference (ISC2), 2016 IEEE International*, pages 1–6. IEEE.

Benjamins, V. R., Casanovas, P., Breuker, J., and Gangemi, A. (2005). Law and the semantic web, an introduction. In *Law and the Semantic Web*, pages 1–17. Springer.

Berendt, B., Dragan, L., Hollink, L., Luczak-Rösch, M., Demidova, E., Dietze, S., Szymanski, J., and Breslin, J. G., editors (2015). *Joint Proceedings of the 5th International Workshop on Using the Web in the Age of Data (USEWOD '15) and the 2nd International Workshop on Dataset PROFIling and fEderated Search for Linked Data (PROFILES '15) co-located with the 12th European Semantic Web Conference (ESWC 2015), Portorož, Slovenia, May 31 - June 1, 2015*, volume 1362 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Berners-Lee, T. (1989). Information Management: A Proposal. Technical report, CERN.

Berners-Lee, T. (1993). Html+. [Online; accessed 24-Nov-2016].

Berners-Lee, T. et al. (1998). Semantic Web road map.

Berners-Lee, T., Fielding, R. T., and Masinter, L. (2005). Uniform resource identifier (uri): Generic syntax. STD 66, RFC Editor. `http://www.rfc-editor.org/rfc/rfc3986.txt`.

Berners-Lee, T., Masinter, L., and McCahill, M. (1994). Uniform Resource Locators (URL). RFC 1738, RFC Editor. `http://www.rfc-editor.org/rfc/rfc1738.txt`.

Biemann, C. (2005). Ontology learning from text: A survey of methods. In *LDV forum*, volume 20-2, pages 75–93.

Bischof, S., Martin, C., Polleres, A., and Schneider, P. (2015). Collecting, Integrating, Enriching and Republishing Open City Data as Linked Data. In *International Semantic Web Conference*, pages 57–75. Springer.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227.

Blomqvist, E., Presutti, V., Daga, E., and Gangemi, A. (2010). Experimenting with eXtreme design. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 120–134. Springer Berlin Heidelberg.

Bohli, J.-M., Skarmeta, A., Victoria Moreno, M., Garcia, D., and Langendorfer, P. (2015). SMARTIE project: Secure IoT data management for smart cities. In *Recent Advances in Internet of Things (RIoT), 2015 International Conference on*, pages 1–6. IEEE.

Bonatti, P. A. and Olmedilla, D. (2007). Rule-based Policy Representation and Reasoning for the Semantic Web. In *Proceedings of the Third International Summer School Conference on Reasoning Web*, RW'07, pages 240–268, Berlin, Heidelberg. Springer-Verlag.

Börger, E. (2012). Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL. *Software & Systems Modeling*, 11(3):305–318.

Börger, E. (2016). Modeling Distributed Algorithms by Abstract State Machines Compared to Petri Nets. In *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z*, pages 3–34. Springer.

Börger, E. and Fleischmann, A. (2015). Abstract state machine nets: closing the gap between business process models and their implementation. In *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management*, page 1. ACM.

Borgman, C. L. (2015). *Big data, little data, no data: Scholarship in the networked world*. Mit Press.

Boros, E., Čepek, O., and Kučera, P. (2013). A decomposition method for CNF minimality proofs. *Theoretical Computer Science*, 510:111–126.

Braden, R. (1989). Requirements for internet hosts - communication layers. STD 3, RFC Editor. `http://www.rfc-editor.org/rfc/rfc1122.txt`.

Brauer, W. and Reisig, W. (2009). Carl Adam Petri and "Petri Nets". *Fundamental Concepts in Computer Science*, 3(5):129.

Breuker, J. et al. (2004). Constructing a legal core ontology: LRI-Core. In *Proceedings WONTO-2004, Workshop on ontologies and their applications*, pages 115–126. IVI (FNWI).

Breuker, J. and Wielinga, B. (1987). KADS - structured knowledge acquisition for expert systems. In *5th International Workshop Vol. 2 on Expert systems & their applications*, pages 887–900. Agence de l'Informatique.

Brewster, C., Ciravegna, F., and Wilks, Y. (2002). User-centred ontology learning for knowledge management. In *International Conference on Application of Natural Language to Information Systems*, pages 203–207. Springer.

Bruns, A. (2008). *Blogs, Wikipedia, Second Life, and beyond: From production to produsage*, volume 45. Peter Lang.

Bruza, P. D. and Van der Weide, T. (1989). *The semantics of data flow diagrams*. University of Nijmegen, Department of Informatics, Faculty of Mathematics and Informatics.

Buneman, P., Khanna, S., and Wang-Chiew, T. (2001). Why and where: A characterization of data provenance. In *International conference on database theory*, pages 316–330. Springer.

Burstein, M., Bussler, C., Finin, T., Huhns, M. N., Paolucci, M., Sheth, A. P., Williams, S., and Zaremba, M. (2005). A semantic web services architecture. *IEEE Internet Computing*, 9(5):72–81.

Burton, A., Koers, H., Manghi, P., La Bruzzo, S., Aryani, A., Diepenbroek, M., and Schindler, U. (2015). On bridging data centers and publishers: the data-literature interlinking service. In *Research Conference on Metadata and Semantics Research*, pages 324–335. Springer.

Bylander, T. and Chandrasekaran, B. (1987). Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies*, 26(2):231–243.

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., and Vo, H. T. (2006). VisTrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747. ACM.

Candela, L., Castelli, D., Manghi, P., and Tani, A. (2015). Data journals: A survey. *Journal of the Association for Information Science and Technology*, 66(9):1747–1762.

Cardellino, C., Villata, S., Gandon, F., Governatori, G., Lam, B., and Rotolo, A. (2014). Licentia: a Tool for Supporting Users in Data Licensing on the Web of Data. In Horridge, M., Rospocher, M., and van Ossenbruggen, J., editors, *Proceedings of the ISWC 2014 Posters & Demonstrations Track, a track within the 13th International Semantic Web Conference (ISWC 2014)*, Riva del Garda, Italy.

Carminati, B., Ferrari, E., and Thuraisingham, B. M. (2004). Using RDF for policy specification and enforcement. In *15th International Workshop on Database and Expert Systems Applications (DEXA 2004), with CD-ROM, 30 August - 3 September 2004, Zaragoza, Spain*, pages 163–167. IEEE Computer Society.

Carothers, G. (2014). RDF 1.1 n-quads. W3C recommendation, W3C. http://www.w3.org/TR/2014/REC-n-quads-20140225/.

Casanovas, P. (2015). Conceptualisation of rights and meta-rule of law for the web of data. *Journal of Governance and Regulation*, 4.

Casanovas, P., Palmirani, M., Peroni, S., van Engers, T., and Vitali, F. (2016a). Semantic web for the legal domain: the next step. *Semantic Web*, 7(3):213–227.

Casanovas, P., Rodríguez-Doncel, V., Santos, C., and Gómez-Pérez, A. (2016b). A European Framework for Regulating Data and Metadata Markets. In *PrivOn@ ISWC*.

Casanovas, P., Sartor, G., Biasiotti, M. A., and Fernández-Barrera, M. (2011). Introduction: Theory and methodology in legal ontology engineering: Experiences and future directions. In *Approaches to Legal Ontologies*, pages 1–14. Springer.

Casanovas, P., Sartor, G., Casellas, N., and Rubino, R. (2008). *Computable models of the law*. Springer.

Catron, B. A. and Ray, S. R. (1991). ALPS: A language for process specification. *International Journal of Computer Integrated Manufacturing*, 4(2):105–113.

Chang, S.-F., Sikora, T., and Purl, A. (2001). Overview of the MPEG-7 standard. *IEEE Transactions on circuits and systems for video technology*, 11(6):688–695.

Chein, M. (1969). Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin mathématique de la Société des Sciences Mathématiques de la République Socialiste de Roumanie*, 13(61):21–25.

Chen-Burger, Y.-H., Tate, A., and Robertson, D. (2002). Enterprise modelling: A declarative approach for FBPML.

Cheney, J. (2013). Semantics of the PROV data model. W3C note, W3C. http://www.w3.org/TR/2013/NOTE-prov-sem-20130430/.

Cheney, J., Chiticariu, L., Tan, W.-C., et al. (2009). Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases*, 1(4):379–474.

Cieri, C. and DiPersio, D. (2015). A License Scheme for a Global Federated Language Service Infrastructure. In *International Workshop on Worldwide Language Service Infrastructure*, pages 86–98. Springer.

Cimiano, P., Hotho, A., and Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res.(JAIR)*, 24(1):305–339.

Cimiano, P., Hotho, A., Stumme, G., and Tane, J. (2004). Conceptual knowledge processing with formal concept analysis and ontologies. In *International Conference on Formal Concept Analysis*, pages 189–207. Springer.

Clancey, W. J. (1985). Heuristic classification. *Artificial intelligence*, 27(3):289–350.

Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., and Little, R. (2002). *Documenting software architectures: views and beyond*. Pearson Education.

Codocedo, V. and Napoli, A. (2015). Formal concept analysis and information retrieval–a survey. In *International Conference on Formal Concept Analysis*, pages 61–77. Springer.

Corcho, O., Fernández-López, M., and Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies. where is their meeting point? *Data & knowledge engineering*, 46(1):41–64.

Corcho, O., Fernández-López, M., Gómez-Pérez, A., and López-Cima, A. (2005). Building legal ontologies with METHONTOLOGY and WebODE. In *Law and the semantic web*, pages 142–157. Springer.

Corcho, Ó., Garijo Verdejo, D., Belhajjame, K., Zhao, J., Missier, P., Newman, D., Palma, R., Bechhofer, S., Garc a Cuesta, E., Gómez-Pérez, J. M., et al. (2012). Workflow-centric research objects: First class citizens in scholarly discourse. In *Proceedings of the ESWC2012 Workshop on the Future of Scholarly Communication in the Semantic Web (SePublica2012)*. Informatica.

Cox, A. M. and Pinfield, S. (2014). Research data management and libraries: Current activities and future priorities. *Journal of Librarianship and Information Science*, 46(4):299–316.

Cox, S. and Little, C. (2016). Time ontology in OWL. W3C working draft, W3C. http://www.w3.org/TR/2016/WD-owl-time-20160712/.

Craig, I. D., Plume, A. M., McVeigh, M. E., Pringle, J., and Amin, M. (2007). Do open access articles have greater citation impact?: a critical review of the literature. *Journal of Informetrics*, 1(3):239–248.

Cui, Y., Widom, J., and Wiener, J. L. (2000). Tracing the lineage of view data in a warehousing environment. *ACM Transactions on Database Systems (TODS)*, 25(2):179–227.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S. (2002). Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *IEEE Internet computing*, 6(2):86–93.

Daga, E., Adamou, A., d'Aquin, M., and Motta, E. (2016a). Addressing Exploitability of Smart City Data. In *International Smart Cities Conference (ISC2)*. IEEE.

Daga, E., Blomqvist, E., Gangemi, A., Montiel, E., Nikitina, N., Presutti, V., and Villazon-Terrazas, B. (2007). D2. 5.2: pattern based ontology design: methodology and software support. Technical report, Tech. rep., NeOn Project.

Daga, E., Blomqvist, E., Gangemi, A., Montiel, E., Nikitina, N., Presutti, V., and Villazón Terrazas, B. (2010). NeOn Deliverable 2.5.2 - Pattern based ontology design: methodology and software support. Technical Report 2.5.2, NeOn Project - Co-funded by the European Commission's Sixth Framework Programme - Grant Number IST-2005-027595.

Daga, E., d'Aquin, M., Gangemi, A., and Motta, E. (2015a). Propagation of Policies in Rich Data Flows. In *Proceedings of the 8th International Conference on Knowledge Capture*, page 5. ACM.

Daga, E., d'Aquin, M., Adamou, A., and Brown, S. (2016b). The Open University Linked Data–data. open. ac. uk. *Semantic Web*, 7(2):183–191.

Daga, E., d'Aquin, M., Gangemi, A., and Motta, E. (2014). Describing Semantic Web Applications Through Relations Between Data Nodes.

Daga, E., d'Aquin, M., Motta, E., and Gangemi, A. (2015b). A Bottom-Up Approach for Licences Classification and Selection. In *Proceedings of the International Workshop on Legal Domain And Semantic Web Applications (LeDA-SWAn) held during the 12th Extended Semantic Web Conference (ESWC 2015)*. CEUR Workshop Proceedings.

Daga, E., Gangemi, A., and Motta, E. (2017 - to appear). Reasoning with Data Flows and Policy Propagation Rules. *Semantic Web Journal*.

Daga, E., Presutti, V., Gangemi, A., and Salvati, A. (2008). http://ontologydesignpatterns. org [ODP]. In *Proceedings of the 2007 International Conference on Posters and Demonstrations-Volume 401*, pages 169–170. CEUR-WS. org.

d'Aquin, M., Adamou, A., Daga, E., Liu, S., Thomas, K., and Motta, E. (2014). Dealing with Diversity in a Smart-City Datahub. In Omitola, T., Breslin, J., and Barnaghi, P., editors, *Proceedings of the Fifth Workshop on Semantics for Smarter Cities, a Workshop at the 13th International Semantic Web Conference (ISWC 2014)*, Riva del Garda, Italy. CEUR-WS.org.

d'Aquin, M., Adamou, A., and Dietze, S. (2013). Assessing the educational linked data landscape. In *Proceedings of the 5th Annual ACM Web Science Conference*, pages 43–46. ACM.

d'Aquin, M., Allocca, C., and Collins, T. (2012). DiscOU: A Flexible Discovery Engine for Open Educational Resources Using Semantic Indexing and Relationship Summaries. In *International Semantic Web Conference (Posters & Demos)*.

d'Aquin, M., Davies, J., and Motta, E. (2015a). Smart cities' data: Challenges and opportunities for semantic technologies. *IEEE Internet Computing*, 19(6):66–70.

d'Aquin, M., Davies, J., and Motta, E. (2015b). Smart Cities' Data: Challenges and Opportunities for Semantic Technologies. *Internet Computing, IEEE*, 19(6):66–70.

d'Aquin, M. and Motta, E. (2016). The Epistemology of Intelligent Semantic Web Systems. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 6(1):1–88.

David, P. A. (2004). Towards a Cyberinfrastructure for Enhanced Scientific Collaboration: Providing its 'Soft' Foundations May Be the Hardest Part. Technical Report 4, Oxford Internet Institute Research Paper Series.

Davies, P. (2012). Can Governments Improve Higher Education Through 'Informing Choice'? *British Journal of Educational Studies*, 60(3):261–276.

Davis, I. (2005). VANN: A vocabulary for annotating vocabulary descriptions. Technical report, Technical report.

De Bruijn, J., Lausen, H., Polleres, A., and Fensel, D. (2006). The web service modeling language WSML: an overview. In *ESWC*, volume 4011, pages 590–604. Springer.

De Roure, D., Goble, C., and Stevens, R. (2007). Designing the myexperiment virtual research environment for the social sharing of workflows. In *e-Science and Grid Computing, IEEE International Conference on*, pages 603–610. IEEE.

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., et al. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237.

Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., and Tonella, P. (2009). Semantically-aided business process modeling. In *International Semantic Web Conference*, pages 114–129. Springer.

Di Penta, M., German, D. M., and Antoniol, G. (2010). Identifying licensing of jar archives using a code-search approach. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, pages 151–160. IEEE.

du Boucher-Ryan, P. and Bridge, D. (2006). Collaborative recommending using formal concept analysis. *Knowledge-Based Systems*, 19(5):309–315.

Duerst, M. and Suignard, M. (2005). Internationalized resource identifiers (iris). RFC 3987, RFC Editor. `http://www.rfc-editor.org/rfc/rfc3987.txt`.

English, M., Auer, S., and Domingue, J. (2016). Block chain technologies & the semantic web: A framework for symbiotic development. In *Computer Science Conference for University of Bonn Students, J. Lehmann, H. Thakkar, L. Halilaj, and R. Asmat, Eds*, pages 47–61.

Erickson, J. and Maali, F. (2014). Data catalog vocabulary (DCAT). W3C recommendation, W3C. http://www.w3.org/TR/2014/REC-vocab-dcat-20140116/.

Erickson, J. S., Rozell, E., Shi, Y., Zheng, J., Ding, L., and Hendler, J. A. (2011). Twc international open government dataset catalog. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 227–229. ACM.

Erl, T. (2004). *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice Hall PTR.

Erné, M., Koslowski, J., Melton, A., and Strecker, G. E. (1993). A primer on Galois connections. *Annals of the New York Academy of Sciences*, 704(1):103–125.

Eschenfelder, K. R. and Johnson, A. (2014). Managing the data commons: controlled sharing of scholarly data. *Journal of the Association for Information Science and Technology*, 65(9):1757–1774.

Fensel, D., Benjamins, V. R., Motta, E., Wielinga, B., et al. (1999). UPML: A framework for knowledge system reuse. In *IJCAI*, pages 16–23.

Fensel, D. and Bussler, C. (2002). The web service modeling framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137.

Fensel, D., Lausen, H., Polleres, A., De Bruijn, J., Stollberg, M., Roman, D., and Domingue, J. (2006). *Enabling semantic web services: the web service modeling ontology*. Springer Science & Business Media.

Fensel, D., Motta, E., Van Harmelen, F., Benjamins, V. R., Crubezy, M., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., Musen, M., et al. (2003). The unified problem-solving method development language UPML. *Knowledge and Information Systems*, 5(1):83–131.

Ferré, S. (2015). A proposal for extending formal concept analysis to knowledge graphs. In *International Conference on Formal Concept Analysis*, pages 271–286. Springer.

Ferreira, D. R., Alves, S., and Thom, L. H. (2011). Ontology-based discovery of workflow activity patterns. In *International Conference on Business Process Management*, pages 314–325. Springer.

Feteris, E. T. (2017). *Fundamentals of legal argumentation: A survey of theories on the justification of judicial decisions*, volume 1. Springer.

Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L., Leach, P. J., and Berners-Lee, T. (1999). Hypertext transfer protocol – http/1.1. RFC 2616, RFC Editor. `http://www.rfc-editor.org/rfc/rfc2616.txt`.

Fielding, R. T. and Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation.

Flavián, C. and Guinalíu, M. (2006). Consumer trust, perceived security and privacy policy: three basic elements of loyalty to a web site. *Industrial Management & Data Systems*, 106(5):601–620.

Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., and Brger, E. (2014). *Subject-oriented business process management*. Springer Publishing Company, Incorporated.

Fortuna, B., Mladenič, D., and Grobelnik, M. (2006). Semi-automatic construction of topic ontologies. In *Semantics, Web and Mining*, pages 121–131. Springer Berlin Heidelberg.

Foukarakis, I. E., Kapitsaki, G. M., and Tselikas, N. D. (2012). Choosing Licenses In Free Open Source Software. In *SEKE*, pages 200–204.

Frosterus, M., Hyvönen, E., and Laitio, J. (2011). Datafinland—a semantic portal for open and linked datasets. In *The Semanic Web: Research and Applications*, pages 243–254. Springer.

Gangemi, A. (2005). Ontology design patterns for semantic web content. In *International semantic web conference*, pages 262–276. Springer.

Gangemi, A., Borgo, S., Catenacci, C., and Lehmann, J. (2004). Task taxonomies for knowledge content. METOKIS Deliverable D7.

Gangemi, A., Daga, E., Salvati, A., Troiani, G., and Baldassarre, C. (2011). Linked Open Data for the Italian PA: the CNR experience. *Informatica e diritto*, 20(1-2):453–476.

Gangemi, A., Gómez-Pérez, A., Presutti, V., and Suárez-Figueroa, M. C. (2007). Towards a catalog of OWL-based ontology design patterns. In *Acta de la XII Conferencia de la Asociación Española para la Inteligencia Artificial*. Informatica.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. (2002). Sweetening ontologies with DOLCE. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 166–181. Springer.

Gangemi, A. and Mika, P. (2003). Understanding the semantic web through descriptions and situations. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 689–706.

Gangemi, A., Peroni, S., Shotton, D., and Vitali, F. (2014). A pattern-based ontology for describing publishing workflows. In *Proceedings of the 5th International Conference on Ontology and Semantic Web Patterns - Volume 1302*, WOP'14, pages 2–13, Aachen, Germany, Germany. CEUR-WS.org.

Gangemi, A., Prisco, A., Sagri, M.-T., Steve, G., and Tiscornia, D. (2003). Some ontological tools to support legal regulatory compliance, with a case study. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 607–620. Springer.

Gangemi, A., Sagri, M.-T., and Tiscornia, D. (2005). A constructive framework for legal ontologies. In *Law and the semantic web*, pages 97–124. Springer.

Ganter, B., Stumme, G., and Wille, R., editors (2005). *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science*. Springer.

García, R., Gil, R., and Delgado, J. (2007). A web ontologies framework for digital rights management. *Artificial Intelligence and Law*, 15(2):137–154.

Garijo, D. (2017). AI buzzwords explained: scientific workflows. *AI Matters*, 3(1):4–8.

Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gil, Y., and Goble, C. (2014). Common motifs in scientific workflows: An empirical analysis. *Future Generation Computer Systems*, 36:338–351.

Garijo, D., Corcho, O., and Gil, Y. (2013). Detecting common scientific workflow fragments using templates and execution provenance. In *Proceedings of the seventh international conference on Knowledge capture*, pages 33–40. ACM.

Garijo, D. and Eckert, K. (2013). Dublin core to PROV mapping. W3C note, W3C. http://www.w3.org/TR/2013/NOTE-prov-dc-20130430/.

Garijo, D. and Gil, Y. (2011). A new approach for publishing workflows: Abstractions, standards, and linked data. In *Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science*, WORKS '11, pages 47–56, New York, NY, USA. ACM.

Gavriloaie, R., Nejdl, W., Olmedilla, D., Seamons, K. E., and Winslett, M. (2004). No Registration Needed: How to Use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web. In *The Semantic Web: Research and Applications*, pages 342–356. Springer.

German, D. M., Di Penta, M., and Davies, J. (2010a). Understanding and auditing the licensing of open source software distributions. In *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*, pages 84–93. IEEE.

German, D. M., Manabe, Y., and Inoue, K. (2010b). A sentence-matching method for automatic license identification of source code files. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 437–446. ACM.

Gibbins, N., Harris, S., and Shadbolt, N. (2004). Agent-based semantic web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(2):141–154.

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., and Myers, J. (2007). Examining the challenges of scientific workflows. *Ieee computer*, 40(12):26–34.

Gil, Y., Ratnakar, V., Kim, J., Gonzalez-Calero, P., Groth, P., Moody, J., and Deelman, E. (2011). Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems*, 26(1):62–72.

Glimm, B., Horridge, M., Parsia, B., Patel, P. F., et al. (2009). A syntax for rules in owl 2. In *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*, volume 529. CEUR.

Goderis, A., Sattler, U., Lord, P., and Goble, C. (2005). Seven bottlenecks to workflow reuse and repurposing. In *International Semantic Web Conference*, pages 323–337. Springer.

Godin, R., Missaoui, R., and Alaoui, H. (1995). Incremental concept formation algorithms based on Galois (concept) lattices. *Computational intelligence*, 11(2):246–267.

Goecks, J., Nekrutenko, A., and Taylor, J. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):1.

Gómez-Pérez, A. and Benjamins, R. (1999). Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. In *IJCAI. CEUR Workshop Proceedings*. IJCAI and the Scandinavian AI Societies.

Gómez-Pérez, J. M. and Corcho, O. (2008). Problem-solving methods for understanding process executions. *Computing in Science & Engineering*, 10(3):47–52.

Gong, S. and Xiong, J. (2009). Interaction mismatch discovery based transformation from BPMN to BPEL. In *Services Computing, 2009. SCC'09. IEEE International Conference on*, pages 292–299. IEEE.

Gordon, T. F. (2008). Constructing Legal Arguments with Rules in the Legal Knowledge Interchange Format (LKIF). *Computable Models of the Law, Languages, Dialogues, Games, Ontologies*, 2008:162–184.

Gordon, T. F. (2011). Analyzing open source license compatibility issues with Carneades. In *Proceedings of the 13th International Conference on Artificial Intelligence and Law*, pages 51–55. ACM.

Governatori, G. and Iannella, R. (2011). A modelling and reasoning framework for social networks policies. *Enterprise Information Systems*, 5(1):145–167.

Governatori, G., Lam, H.-P., Rotolo, A., Villata, S., Atemezing, G., and Gandon, F. (2014). Checking licenses compatibility between vocabularies and data. In *Proceedings of the Fifth International Workshop on Consuming Linked Data (COLD2014)*.

Governatori, G., Lam, H.-P., Rotolo, A., Villata, S., and Gandon, F. (2013a). Heuristics for Licenses Composition. In *JURIX*, pages 77–86.

Governatori, G., Rotolo, A., Villata, S., and Gandon, F. (2013b). One License to Compose Them All. In *The Semantic Web–ISWC 2013*, pages 151–166. Springer.

Gray, P. N., Brookfield, V., and McJohn, S. M. (1998). Artificial Legal Intelligence. *Harvard Journal of Law & Technology*, 12(1).

Green, T. J., Karvounarakis, G., and Tannen, V. (2007). Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40. ACM.

Greenberg, J. (2005). Understanding metadata and metadata schemes. *Cataloging & classification quarterly*, 40(3-4):17–36.

Grewe, N. (2010). A generic reification strategy for n-ary relations in DL. In *OBML 2010 Workshop Proceedings*.

Griffo, C., Almeida, J. P. A., and Guizzardi, G. (2015). A Systematic Mapping of the Literature on Legal Core Ontologies. In *ONTOBRAS*.

Groth, P. (2007). *The origin of data: Enabling the determination of provenance in multi-institutional scientific systems through the documentation of processes*. PhD thesis, University of Southampton.

Gruber, T. R. (1991). The role of common ontology in achieving sharable, reusable knowledge bases. *Principles of Knowledge Representation and Reasoning*, 91:601–602.

Grüninger, M. (2004). Ontology of the process specification language. In *Handbook on ontologies*, pages 575–592. Springer.

Grüninger, M. and Fox, M. S. (1995). Methodology for the design and evaluation of ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing*. International Joint Conference on Artificial Intelligence, Citeseer.

Gruninger, M. and Menzel, C. (2003). The process specification language (PSL) theory and applications. *AI magazine*, 24(3):63.

Guarino, N. (1994). The ontological level. In *PHILOSOPHY AND THE COGNITIVE SCIENCES*, pages 443–456. Holder-Pichler-Tempsky.

Guarino, N. et al. (1998). Formal ontology and information systems. In *Proceedings of FOIS*, volume 98, pages 81–97.

Guha, R. and Brickley, D. (2014). RDF schema 1.1. W3C recommendation, W3C. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

Guinard, D. and Trifa, V. (2009). Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*, page 15.

Guinard, D., Trifa, V., Mattern, F., and Wilde, E. (2011). From the internet of things to the web of things: Resource-oriented architecture and best practices. In *Architecting the Internet of Things*, pages 97–129. Springer.

Gurgen, L., Gunalp, O., Benazzouz, Y., and Gallissot, M. (2013). Self-aware cyber-physical systems and applications in smart buildings and cities. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1149–1154. EDA Consortium.

Hammar, K. and Presutti, V. (2016). Template-Based Content ODP Instantiation. In *Proceedings of the Workshop on Ontology and Semantic Web Patterns (7th Edition) - WOP2016*.

Hammer, P. L. and Kogan, A. (1993). Optimal compression of propositional Horn knowledge bases: complexity and approximation. *Artificial Intelligence*, 64(1):131–145.

Handfield, R. B. and Nichols, E. L. (1999). *Introduction to supply chain management*, volume 1. prentice Hall Upper Saddle River, NJ.

Hartig, O. (2009). Querying trust in RDF data with tSPARQL. In Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., and Simperl, E. P. B., editors, *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings*, volume 5554 of *Lecture Notes in Computer Science*, pages 5–20. Springer.

Hartig, O. and Zhao, J. (2010). Publishing and consuming provenance metadata on the web of linked data. In *Provenance and Annotation of Data and Processes*, pages 78–90. Springer.

Haslhofer, B. and Isaac, A. (2011). data. europeana. eu: The europeana linked open data pilot. In *International Conference on Dublin Core and Metadata Applications*, pages 94–104.

Havey, M. (2005). *Essential business process modeling*. " O'Reilly Media, Inc.".

Heath, T. and Bizer, C. (2011). Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136.

Hendler, J. and Berners-Lee, T. (2010). From the semantic web to social machines: A research challenge for ai on the world wide web. *Artificial Intelligence*, 174(2):156–161.

Hepp, M., Leymann, F., Domingue, J., Wahler, A., and Fensel, D. (2005). Semantic business process management: A vision towards using semantic web services for business process management. In *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on*, pages 535–540. IEEE.

Hepp, M. and Roman, D. (2007). An ontology framework for semantic business process management. *Wirtschaftinformatik Proceedings 2007*, page 27.

High, R. (2012). The era of cognitive systems: An inside look at ibm watson and how it works. *IBM Corporation, Redbooks*.

Hitzler, P., Krotzsch, M., and Rudolph, S. (2009). *Foundations of semantic web technologies*. CRC press.

Hoekstra, R., Breuker, J., Di Bello, M., Boer, A., et al. (2007). The LKIF Core Ontology of Basic Legal Concepts. *LOAIT*, 321:43–63.

Hollingsworth, D. and Hampshire, U. (1995). Workflow management coalition: The workflow reference model. *Document Number TC00-1003*, 19.

Hunter, J. (2001). Adding multimedia to the semantic web: Building an mpeg-7 ontology. In *Proceedings of the First International Conference on Semantic Web Working*, pages 261–283. CEUR-WS. org.

Hunter, J. (2003). Enhancing the semantic interoperability of multimedia through a core ontology. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(1):49–58.

Iannella, R. (2001). Open digital rights management. In *World Wide Web Consortium (W3C) DRM Workshop*.

Iannella, R. (2002). Open digital rights language (ODRL) version 1.1. *W3c Note*.

Iannella, R., Guth, S., Pähler, D., and Kasten, A. (2012). ODRL: Open digital rights language 2.1. Technical report, W3C ODRL Community Group.

Iannella, R., Guth, S., Pähler, D., and Kasten, A. (2015). ODRL: Open Digital Rights Language 2.1. Technical report, W3C Community Group.

ISO (1998). ISO 3166-2:1998 Codes for the representation of names of countries and their subdivisions — Part 2: Country subdivision code. Technical report, International Organization for Standardization, Geneva, Switzerland.

Jamkhedkar, P. A. and Heileman, G. L. (2009). Rights Expression Languages. *Handbook of Research on Secure Multimedia Distribution. New York: IGI Books*, pages 1–21.

Janssen, M., Kuk, G., and Wagenaar, R. W. (2008). A survey of Web-based business models for e-government in the Netherlands. *Government Information Quarterly*, 25(2):202–220.

Jasanoff, S. (2011). *Designs on nature: Science and democracy in Europe and the United States*. Princeton University Press.

Javanmardi, S., Amini, M., Jalili, R., and GanjiSaffar, Y. (2006). SBAC: "A Semantic-Based Access Control Model". In *11th Nordic Workshop on Secure IT-systems (NordSec'06), Linkping, Sweden*, volume 22.

Jiang, D., Pierre, G., and Chi, C.-H. (2010). Autonomous resource provisioning for multi-service web applications. In *Proceedings of the 19th international conference on World wide web*, pages 471–480. ACM.

Jicheng, W., Yuan, H., Gangshan, W., and Fuyan, Z. (1999). Web mining: knowledge discovery on the Web. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 2, pages 137–141. IEEE.

Johnston, W. M., Hanna, J., and Millar, R. J. (2004). Advances in dataflow programming languages. *ACM Computing Surveys (CSUR)*, 36(1):1–34.

Kagal, L., Finin, T., and Joshi, A. (2003). A policy based approach to security for the semantic web. In *International Semantic Web Conference*, pages 402–418. Springer.

Kahn, G. and MacQueen, D. (1976). Coroutines and networks of parallel processes. Technical Report inria-00306565, INRIA.

Kapitsaki, G. M., Tselikas, N. D., and Foukarakis, I. E. (2015). An insight into license tools for open source software systems. *Journal of Systems and Software*, 102:72–87.

Kaplan, A. M. and Haenlein, M. (2010). Users of the world, unite! The challenges and opportunities of Social Media. *Business horizons*, 53(1):59–68.

Kavi, K. M., Buckles, B. P., and Bhat, U. N. (1986). A formal definition of data flow graph models. *IEEE Transactions on computers*, 35(11):940–948.

Kawase, R., Fisichella, M., Niemann, K., Pitsilis, V., Vidalis, A., Holtkamp, P., and Nunes, B. (2013). OpenScout: Harvesting Business and Management Learning Objects from the Web of Data. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 445–450. ACM.

Khrouf, H. and Troncy, R. (2016 (Submitted in 2012)). EventMedia: A LOD dataset of events illustrated with media. *Semantic Web journal, Special Issue on Linked Dataset descriptions*, 7(2):193–199.

Kietzmann, J. H., Hermkens, K., McCarthy, I. P., and Silvestre, B. S. (2011). Social media? Get serious! Understanding the functional building blocks of social media. *Business horizons*, 54(3):241–251.

Kitchin, R. (2014). The real-time city? Big data and smart urbanism. *GeoJournal*, 79(1):1–14.

Kiškis, M. and Petrauskas, R. (2004). Ict adoption in the judiciary: classifying of judicial information. *International Review of Law, Computers & Technology*, 18(1):37–45.

Kloeckl, K., Senn, O., Di Lorenzo, G., and Ratti, C. (2011). Live singapore!-an urban platform for real-time data to program the city. *Proceedings of Computers in Urban Planning and Urban Management (CUPUM), Alberta, Canada*.

Klusch, M., Kapahnke, P., Schulte, S., Lecue, F., and Bernstein, A. (2016). Semantic web service search: a brief survey. *KI-Künstliche Intelligenz*, 30(2):139–147.

Knutilla, A., Schlenoff, C., Ray, S., Polyak, S. T., Tate, A., Cheah, S. C., and Anderson, R. C. (1998). Process specification language: An analysis of existing representations. *National Institute of Standards and Technology (NIST), Gaithersburg (MD), NISTIT*, 6160.

Ko, R. K., Lee, S. S., and Wah Lee, E. (2009). Business process management (BPM) standards: a survey. *Business Process Management Journal*, 15(5):744–791.

Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., and Lee, R. (2009). Media Meets Semantic Web - How the BBC Uses DBpedia and Linked Data to Make Connections. In *European Semantic Web Conference*, pages 723–737. Springer.

Kondylakis, H., Doerr, M., and Plexousakis, D. (2009). Empowering provenance in data integration. In Grundspenkis, J., Morzy, T., and Vossen, G., editors, *Advances in Databases and Information Systems, 13th East European Conference, ADBIS 2009, Riga, Latvia, September 7-10, 2009. Proceedings*, volume 5739 of *Lecture Notes in Computer Science*, pages 270–285. Springer.

Kopeckỳ, J., Vitvar, T., Bournez, C., and Farrell, J. (2007). Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6).

Kotoulas, S., Lopez, V., Lloyd, R., Sbodio, M. L., Lecue, F., Stephenson, M., Daly, E., Bicer, V., Gkoulalas-Divanis, A., Di Lorenzo, G., et al. (2014). SPUD—Semantic Processing of Urban Data. *Web Semantics: Science, Services and Agents on the World Wide Web*.

Krumm, J., Davies, N., and Narayanaswami, C. (2008). User-generated content. *IEEE Pervasive Computing*, 4(7):10–11.

Kuznetsov, S. O. and Obiedkov, S. A. (2002). Comparing performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):189–216.

Lam, H.-P. and Governatori, G. (2009). The making of SPINdle. In *Rule Interchange and Applications*, pages 315–322. Springer.

Lampe, U., Schulte, S., Siebenhaar, M., Schuller, D., and Steinmetz, R. (2010). Adaptive matchmaking for RESTful services based on hRESTS and MicroWSMO. In *Proceedings of the 5th International Workshop on Enhanced Web Service Technologies*, pages 10–17. ACM.

Lanthaler, M., Cyganiak, R., and Wood, D. (2014). RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

Latif, A., Saeed, A. U., Höfler, P., Stocker, A., and Wagner, C. (2009). The Linked Data Value Chain: A Lightweight Model for Business Engineers. In *I-SEMANTICS*, pages 568–575. Citeseer.

Lea, R. and Blackstock, M. (2014). City Hub: A cloud-based IoT Platform for Smart Cities. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 799–804. IEEE.

Lebo, T., McGuinness, D., and Sahoo, S. (2013). PROV-o: The PROV ontology. W3C recommendation, W3C. http://www.w3.org/TR/2013/REC-prov-o-20130430/.

Lee, E. A., Hartmann, B., Kubiatowicz, J., Rosing, T. S., Wawrzynek, J., Wessel, D., Rabaey, J. M., Pister, K., Sangiovanni-Vincentelli, A. L., Seshia, S. A., et al. (2014). The Swarm at the Edge of the Cloud. *IEEE Design & Test*, 31(3):8–20.

Lee, J., Yost, G., Group, P. W., et al. (1996). The pif process interchange format and framework. *MIT Center for Coordination Science, Working Paper*, 194:1996.

Lemieux, V. L. et al. (2016). Provenance: Past, Present and Future in Interdisciplinary and Multidisciplinary Perspective. In *Building Trust in Information*, pages 3–45. Springer.

Lenhart, A., Purcell, K., Smith, A., and Zickuhr, K. (2010). Social Media & Mobile Internet Use among Teens and Young Adults. Millennials. *Pew internet & American life project*.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM.

Li, H., Zhang, X., Wu, H., and Qu, Y. (2005). Design and application of rule based access control policies. In *Proc of the Semantic Web and Policy Workshop*, pages 34–41.

Li, X. and Murata, T. (2010). A knowledge-based recommendation model utilizing formal concept analysis and association. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 4, pages 221–226. IEEE.

Li, Z., Zhou, M., and Wu, N. (2008). A survey and comparison of Petri net-based deadlock prevention policies for flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):173–188.

Litman, J. (1990). Public domain, the. *Emory Lj*, 39:965.

Liu, J., Pacitti, E., Valduriez, P., and Mattoso, M. (2015). A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, 13(4):457–493.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., and Zhao, Y. (2006). Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065.

Lynch, C. (1997). Searching the Internet. *Scientific American*, 276(3):52–56.

Lyon, L. (2012). The informatics transform: Re-engineering libraries for the data decade. *International Journal of Digital Curation*, 7(1):126–138.

Maali, F., Cyganiak, R., and Peristeras, V. (2010). Enabling interoperability of government data catalogues. In Wimmer, M., Chappelet, J., Janssen, M., and Scholl, H. J., editors, *Electronic Government, 9th IFIP WG 8.5 International Conference, EGOV 2010, Lausanne, Switzerland, August 29 - September 2, 2010. Proceedings*, volume 6228 of *Lecture Notes in Computer Science*, pages 339–350. Springer.

Maedche, A. and Staab, S. (2000). Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th international conference on software engineering and knowledge engineering*, pages 231–239. Citeseer.

Maker, R., Cullinane, T., deWitte, P., Knappenberger, W., Perakath, B., and Wells, M. (1992). IDEF3–process description capture method report. *Information Integration for Concurrent Engineering (IICE), Armstrong Laboratory, Wright-Patterson AFB, OH*.

Manabe, Y., Hayase, Y., and Inoue, K. (2010). Evolutional analysis of licenses in FOSS. In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, pages 83–87. ACM.

Manghi, P., Bolikowski, L., Manold, N., Schirrwagen, J., and Smith, T. (2012). Openaireplus: the european scholarly communication data infrastructure. *D-Lib Magazine*, 18(9):1.

Mannocci, A. (2017). *Data Flow Quality Monitoring in Data Infrastructures*. PhD thesis, Ingegneria dell'Informazione, Universita' di Pisa.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). OWL-S: Semantic markup for web services. *W3C member submission*, 22:2007–04.

Martin, M., Kaltenböck, M., Nagy, H., and Auer, S. (2011). The Open Government Data Stakeholder Survey. In *OKCon*.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. (2003). Ontology Library (final). WonderWeb Project Deliverable D18.

Matsumura, F., Kobayashi, I., Kato, F., Kamura, T., Ohmukai, I., and Takeda, H. (2012). Producing and Consuming Linked Open Data on Art with a Local Community. In *COLD*.

McDermott, J. (1988). Preliminary steps toward a taxonomy of problem-solving methods. In *Automating knowledge acquisition for expert systems*, pages 225–256. Springer.

McIlraith, S. A., Son, T. C., and Zeng, H. (2001). Semantic web services. *IEEE intelligent systems*, 16(2):46–53.

Mendling, J. and Nüttgens, M. (2006). EPC markup language (EPML): an XML-based interchange format for event-driven process chains (EPC). *Information Systems and e-Business Management*, 4(3):245–263.

Minami, Y., Takeda, H., Kato, F., Ohmukai, I., Arai, N., Jinbo, U., Ito, M., Kobayashi, S., and Kawamoto, S. (2013). Towards a Data Hub for Biodiversity with LOD. In *Semantic Technology*, pages 356–361. Springer.

Missier, P. (2016). Data trajectories: tracking reuse of published data for transitive credit attribution. *International Journal of Digital Curation*, 11(1):1–16.

Missier, P., Belhajjame, K., and Cheney, J. (2013). The W3C PROV family of specifications for modelling provenance metadata. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 773–776. ACM.

Missier, P. and Moreau, L. (2013). PROV-n: The provenance notation. W3C recommendation, W3C. http://www.w3.org/TR/2013/REC-prov-n-20130430/.

Mladenić, D., Grobelnik, M., Fortuna, B., and Grćar, M. (2009). *Knowledge discovery for semantic web*. Springer.

Moghaddam, M. and Davis, J. G. (2014). Service selection in web service composition: A comparative review of existing approaches. In *Web Services Foundations*, pages 321–346. Springer.

Moniruzzaman, A. and Hossain, D. S. A. (2013). Comparative Study on Agile software development methodologies. *arXiv preprint arXiv:1307.3356*.

Moreau, L. (2010). The foundations for provenance on the web. *Foundations and Trends in Web Science*, 2(2–3):99–241.

Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., and Paulson, P. (2008). The open provenance model: An overview. *Provenance and Annotation of Data and Processes*, pages 323–326.

Moreau, L. and Groth, P. (2013). PROV-overview. W3C note, W3C. http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/.

Moreau, L., Groth, P., Cheney, J., Lebo, T., and Miles, S. (2015). The rationale of PROV. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:235–257.

Moreau, L. and Lebo, T. (2013). Linking across Provenance Bundles. Technical report, World Wide Web Consortium.

Moreau, L. and Missier, P. (2013). PROV-dm: The PROV data model. W3C recommendation, W3C. http://www.w3.org/TR/2013/REC-prov-dm-20130430/.

Moreau, L., Missier, P., and Cheney, J. (2013). Constraints of the PROV data model. W3C recommendation, W3C. http://www.w3.org/TR/2013/REC-prov-constraints-20130430/.

Motta, E. (1997). Trends in knowledge modelling: report on the 7th KEML Workshop. *The Knowledge Engineering Review*, 12(2):209–217.

Motta, E. (1998). An overview of the OCML modelling language. In *the 8th Workshop on Methods and Languages*.

Motta, E., Rajan, T., and Eisenstadt, M. (1990). Knowledge acquisition as a process of model refinement. *Knowledge acquisition*, 2(1):21–49.

Motta, E. and Wiedenbeck, S. (2013). International journal of human-computer studies.

Musetti, A., Nuzzolese, A. G., Draicchio, F., Presutti, V., Blomqvist, E., Gangemi, A., and Ciancarini, P. (2012). Aemoo: Exploratory search based on knowledge patterns over the semantic web. *Semantic Web Challenge*.

Nacer, H. and Aissani, D. (2014). Semantic web services: Standards, applications, challenges and solutions. *Journal of Network and Computer Applications*, 44:134–151.

Nam, T. and Pardo, T. A. (2011). Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*, pages 282–291. ACM.

Naphade, M., Banavar, G., Harrison, C., Paraszczak, J., and Morris, R. (2011). Smarter cities and their innovation challenges. *Computer*, 44(6):32–39.

Nasr, A. and Keshtiarast, A. (2013). Datahub for AURIN and ANDS Project. *Spatial Data Access and Integration To Support Liveability*, page 75.

Navigli, R., Velardi, P., and Faralli, S. (2011). A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI*, volume 11, pages 1872–1877.

Newell, A. (1982). The knowledge level. *Artificial intelligence*, 18(1):87–127.

Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9. ACM.

Norton, B., Pedrinaci, C., Domingue, J., and Zaremba, M. (2008). Semantic execution environments for semantics-enabled SOA. *IT-MUNCHEN-*, 50(2):118.

Obitko, M., Snasel, V., Smid, J., and Snasel, V. (2004). Ontology Design with Formal Concept Analysis. In *CLA*, volume 110.

Ogasawara, E., Dias, J., Silva, V., Chirigati, F., Oliveira, D., Porto, F., Valduriez, P., and Mattoso, M. (2013). Chiron: a parallel engine for algebraic scientific workflows. *Concurrency and Computation: Practice and Experience*, 25(16):2327–2341.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., Wipat, A., et al. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054.

Olabarriaga, S., Pierantoni, G., Taffoni, G., Sciacca, E., Jaghoori, M., Korkhov, V., Castelli, G., Vuerli, C., Becciani, U., Carley, E., et al. (2014). Scientific Workflow Management–For Whom? In *e-Science (e-Science), 2014 IEEE 10th International Conference on*, volume 1, pages 298–305. IEEE.

OMG, S. and Notation, O. (2008). Software & Systems Process Engineering Meta-Model Specification. *OMG Std., Rev*, 2.

Omitola, T., Gibbins, N., and Shadbolt, N. (2010). Provenance in linked data integration. In *Future Internet Assembly*. Event Dates: 16-17 December 2010.

Oren, E. (2008). *Algorithms and components for application development on the semantic web*. PhD thesis, Citeseer.

Osborne, F., Motta, E., and Mulholland, P. (2013). Exploring Scholarly Data with Rexplore. In *The Semantic Web–ISWC 2013*, pages 460–477. Springer.

Pan, G., Qi, G., Zhang, W., Li, S., Wu, Z., and Yang, L. T. (2013). Trace analysis and mining for smart cities: issues, methods, and applications. *IEEE Communications Magazine*, 51(6):120–126.

Papadopoulos, S., Kompatsiaris, Y., Vakali, A., and Spyridonos, P. (2012). Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3):515–554.

Partridge, D. and Wilks, Y. (1987). Does AI have a methodology which is different from software engineering? *Artificial intelligence review*, 1(2):111–120.

Passant, A. (2010). DBRec—music recommendations using DBpedia. In *The Semantic Web–ISWC 2010*, pages 209–224. Springer.

Pazienza, M. T., Pennacchiotti, M., and Zanzotto, F. M. (2005). Terminology extraction: an analysis of linguistic and statistical approaches. In *Knowledge mining*, pages 255–279. Springer.

Pedrinaci, C., Domingue, J., and Sheth, A. P. (2011). Semantic Web Services. In *Handbook of semantic web technologies*, pages 977–1035. Springer.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77.

Peled, A. (2011). When transparency and collaboration collide: The USA open data program. *Journal of the Association for Information Science and Technology*, 62(11):2085–2094.

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, 25(1):81–93.

Peroni, S. and Vitali, F. (2009). Annotations with earmark for arbitrary, overlapping and out-of order markup. In *Proceedings of the 9th ACM symposium on Document engineering*, pages 171–180. ACM.

Petri, C. A. (1962). *Kommunikation mit Automaten*. PhD thesis, Universität Hamburg.

Pfeiffer, S., Navara, E. D., Leithead, T., Hickson, I., O'Connor, T., Faulkner, S., and Berjon, R. (2014). HTML5. W3C recommendation, W3C. http://www.w3.org/TR/2014/REC-html5-20141028/.

Poelmans, J., Elzinga, P., Viaene, S., and Dedene, G. (2010). Formal concept analysis in knowledge discovery: a survey. In *International Conference on Conceptual Structures*, pages 139–153. Springer.

Poelmans, J., Ignatov, D. I., Kuznetsov, S. O., and Dedene, G. (2013a). Formal concept analysis in knowledge processing: A survey on applications. *Expert systems with applications*, 40(16):6538–6560.

Poelmans, J., Kuznetsov, S. O., Ignatov, D. I., and Dedene, G. (2013b). Formal concept analysis in knowledge processing: A survey on models and techniques. *Expert systems with applications*, 40(16):6601–6623.

Polyak, S. T. and Tate, A. (1998). *A Common Process Ontology for Process-Centred Organisations*. Department of Artificial Intelligence, University of Edinburgh.

Posada, J., Toro, C., Wundrak, S., and Stork, A. (2005). Ontology supported semantic simplification of large data sets of industrial plant CAD models for design review visualization. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 184–190. Springer.

Preist, C. (2004). A conceptual architecture for semantic web services. In *International Semantic Web Conference*, pages 395–409. Springer.

Presutti, V., Blomqvist, E., Daga, E., and Gangemi, A. (2012). Pattern-based ontology design. In *Ontology Engineering in a Networked World*, pages 35–64. Springer.

Presutti, V., Daga, E., Gangemi, A., and Blomqvist, E. (2009). eXtreme design with content ontology design patterns. In *Proceedings of the 2009 International Conference on Ontology Patterns-Volume 516*, pages 83–97. CEUR-WS. org.

Priss, U. (2006). Formal concept analysis in information science. *Arist*, 40(1):521–543.

Pryor, G. (2009). Multi-scale data sharing in the life sciences: some lessons for policy makers. *International Journal of Digital Curation*, 4(3):71–82.

Pucella, R. and Weissman, V. (2006). A formal foundation for ODRL. *arXiv preprint cs/0601085*.

Qiu, J., Chen, J., and Wang, Z. (2004). An analysis of backlink counts and web Impact Factors for Chinese university websites. *Scientometrics*, 60(3):463–473.

Qu, Y., Zhang, X., and Li, H. (2004). OREL: an ontology-based rights expression language. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 324–325. ACM.

Ram, S. and Liu, J. (2009). A new perspective on semantics of data provenance. In Freire, J., Missier, P., and Sahoo, S. S., editors, *Proceedings of the First International Workshop on the role of Semantic Web in Provenance Management (SWPM 2009)*, volume 526. CEUR-WS.org.

Rao, J. and Su, X. (2004). A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer.

Recker, J. and Mendling, J. (2007). Lost in business process model translations: how a structured approach helps to identify conceptual mismatch. In *Research Issues in Systems Analysis and Design, Databases and Software Development*, pages 227–259. IGI Global.

Recker, J. C. and Mendling, J. (2006). On the translation between BPMN and BPEL: Conceptual mismatch between process modeling languages. In *The 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium*, pages 521–532. Namur University Press.

Reichman, J. H. and Uhlir, P. F. (2003). A contractually reconstructed research commons for scientific data in a highly protectionist intellectual property environment. *Law and Contemporary problems*, 66(1/2):315–462.

Renfrew, K., Baird, H., Green, H., Davies, P., Hughes, A., Mangan, J., and Slack, K. (2010). Understanding the information needs of users of public information about higher education. Technical report, Higher Education Funding Council for England (HEFCE).

Reynolds, D. and Cyganiak, R. (2014a). The RDF data cube vocabulary. W3C recommendation, W3C. http://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/.

Reynolds, D. and Cyganiak, R. (2014b). The RDF data cube vocabulary. W3C recommendation, W3C. http://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/.

Rizzo, G., Troncy, R., Hellmann, S., and Bruemmer, M. (2012). NERD meets NIF: Lifting NLP Extraction Results to the Linked Data Cloud. *LDOW*, 937.

Rodrigues, T., Benevenuto, F., Cha, M., Gummadi, K., and Almeida, V. (2011). On word-of-mouth based discovery of the web. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 381–396. ACM.

Rodriguez-Doncel, V., Gómez-Pérez, A., and Mihindukulasooriya, N. (2013). Rights declaration in linked data. In *Proceedings of the Fourth International Conference on Consuming Linked Data-Volume 1034*, pages 158–169. CEUR-WS. org.

Rodríguez-Doncel, V., Villata, S., and Gómez-Pérez, A. (2014). A dataset of RDF licenses. In Hoekstra, R., editor, *Legal Knowledge and Information Systems. JURIX 2014: The Twenty-Seventh Annual Conference*. IOS Press.

Roman, D., Keller, U., Lausen, H., De Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web service modeling ontology. *Applied ontology*, 1(1):77–106.

Rotolo, A., Villata, S., and Gandon, F. (2013). A deontic logic semantics for licenses composition in the web of data. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, pages 111–120. ACM.

Ruiz-Rube, I., Dodero, J. M., Palomo-Duarte, M., Ruiz, M., and Gawn, D. (2013). Uses and applications of Software & Systems Process Engineering Meta-Model process models. A systematic mapping study. *Journal of Software: Evolution and Process*.

Russell, N., Ter Hofstede, A. H., Edmond, D., and van der Aalst, W. M. (2004). Workflow data patterns. Technical Report FIT-TR-2004-01, Queensland University of Technology, Brisbane.

Russell, N., van der Aalst, W. M., Ter Hofstede, A. H., and Wohed, P. (2006). On the suitability of UML 2.0 activity diagrams for business process modelling. In *Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling-Volume 53*, pages 95–104. Australian Computer Society, Inc.

Scholz-Reiter, B. and Stickel, E. (2012). *Business process modelling*. Springer Science & Business Media.

Schreiber, G. (2000). *Knowledge engineering and management: the CommonKADS methodology*. MIT press.

Schreiber, G. (2013). Knowledge acquisition and the web. *International Journal of Human-Computer Studies*, 71(2):206–210.

Scott, D. and Sharp, R. (2002). Abstracting application-level web security. In *Proceedings of the 11th international conference on World Wide Web*, pages 396–407. ACM.

Seibt, J. (2001). Formal process ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 333–345. ACM.

Sensoy, M., Norman, T. J., Vasconcelos, W. W., and Sycara, K. (2012). OWL-POLAR: A framework for semantic policy representation and reasoning. *Web Semantics: Science, Services and Agents on the World Wide Web*, 12:148–160.

Shaw, R., Troncy, R., and Hardman, L. (2009). Lode: Linking open descriptions of events. In *The Semantic Web*, pages 153–167. Springer.

Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition: A decade's overview. *Information Sciences*, 280:218–238.

Sheth, A. P., Gomadam, K., and Lathem, J. (2007). SA-REST: Semantically interoperable and easier-to-use services and mashups. *IEEE Internet Computing*, 11(6).

Shields, M. and Taylor, I. (2004). Programming scientific and distributed workflow with Triana services. In *Proceedings of Workflow in Grid Systems Workshop in GGF10*.

Shirky, C. (2011). The political power of social media: Technology, the public sphere, and political change. *Foreign affairs*, pages 28–41.

Shukair, G., Loutas, N., Peristeras, V., and Sklarß, S. (2013). Towards semantically interoperable metadata repositories: The Asset Description Metadata Schema. *Computers in Industry*, 64(1):10–18.

Simmhan, Y., Plale, B., and Gannon, D. (2005a). A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36.

Simmhan, Y. L., Plale, B., and Gannon, D. (2005b). A survey of data provenance in e-science. *ACM Sigmod Record*, 34(3):31–36.

Staab, S. and Studer, R. (2013). *Handbook on ontologies*. Springer Science & Business Media.

Steels, L. (1990). Components of expertise. *AI magazine*, 11(2):28.

Stefik, M. (2014). *Introduction to Knowledge Systems*. Morgan Kaufmann.

Steidl, M., Iannella, R., Rodríguez-Doncel, V., and Myles, S. (2018). ODRL vocabulary & expression 2.2. W3C recommendation, W3C. https://www.w3.org/TR/2018/REC-odrl-vocab-20180215/.

Steyskal, S. and Polleres, A. (2015). Towards Formal Semantics for ODRL Policies. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pages 360–375. Springer.

Stodden, V. (2009). The legal framework for reproducible scientific research: Licensing and copyright. *Computing in Science & Engineering*, 11(1):35–40.

Strunk, A. (2010). QoS-aware service composition: A survey. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 67–74. IEEE.

Studer, R., Benjamins, V., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1):161 – 197.

Studer, R., Grimm, S., and Abecker, A. (2007). *Semantic web services*. Springer.

Suárez-Figueroa, M. C., Gomez-Perez, A., and Fernandez-Lopez, M. (2012). The NeOn methodology for ontology engineering. In *Ontology engineering in a networked world*, pages 9–34. Springer.

Sure, Y., Staab, S., and Studer, R. (2009). Ontology engineering methodology. In *Handbook on ontologies*, pages 135–152. Springer.

Sure, Y., Tempich, C., and Vrandecic, D. (2006). Ontology engineering methodologies. *Semantic Web Technologies: Trends and Research in Ontology-based Systems*, pages 171–190.

Sweeney, S. (2008). The ambiguous origins of the archival principle of" provenance". *Libraries & the Cultural Record*, 43(2):193–213.

Tate, A. (1998). Roots of SPAR—shared planning and activity representation. *The Knowledge Engineering Review*, 13(01):121–128.

Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M. (2014). *Workflows for e-Science: scientific workflows for grids*. Springer Publishing Company, Incorporated.

ter Hofstede, A. H., van der Aalst, W. M., Adams, M., and Russell, N. (2009). *Modern Business Process Automation: YAWL and its support environment*. Springer Science & Business Media.

The W3C SPARQL Working Group (2013). SPARQL 1.1 overview. W3C recommendation, W3C. http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

Thong, W. J. and Ameedeen, M. (2015). A survey of Petri net tools. In *Advanced Computer and Communication Engineering Technology*, pages 537–551. Springer.

Toni, F. (2008). Assumption-Based Argumentation for Epistemic and Practical Reasoning. *Computable Models of the Law, Languages, Dialogues, Games, Ontologies*, 4884:185–202.

Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., and Uszok, A. (2003). Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. In *International Semantic Web Conference*, pages 419–437. Springer.

Townsend, A. M. (2013). *Smart cities: Big data, civic hackers, and the quest for a new utopia*. WW Norton & Company.

Trujillo, J. and Luján-Mora, S. (2003). A UML based approach for modeling ETL processes in data warehouses. In *International Conference on Conceptual Modeling*, pages 307–320. Springer.

Tseng, V. S. and Lin, K. W. (2006). Efficient mining and prediction of user behavior patterns in mobile web systems. *Information and software technology*, 48(6):357–369.

Tuunanen, T., Koskinen, J., and Kärkkäinen, T. (2009). Automated software license analysis. *Automated Software Engineering*, 16(3-4):455–490.

Unicode Consortium and others (2004). The Unicode Standard, Version 4.0. 1, defined by: The Unicode Standard, Version 4.0 (Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1), as amended by Unicode 4.0. 1 (http://www. unicode. org/versions/Unicode4. 0.1/).

Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(02):93–136.

van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., and Barros, A. P. (2003). Workflow patterns. *Distributed and parallel databases*, 14(1):5–51.

Van Der Vet, P. E. and Mars, N. J. (1998). Bottom-up construction of ontologies. *IEEE Transactions on Knowledge and data Engineering*, 10(4):513–526.

van Harmelen, F., ten Teije, A., and Wache, H. (2009). Knowledge engineering rediscovered: towards reasoning patterns for the semantic web. In *Proceedings of the fifth international conference on Knowledge capture*, pages 81–88. ACM.

Vandenbussche, P.-Y. and Vatant, B. (2011). Metadata recommendations for linked open data vocabularies. *Version*, 1:2011–12.

Vardigan, M., Heus, P., and Thomas, W. (2008). Data documentation initiative: Toward a standard for the social sciences. *International Journal of Digital Curation*, 3(1):107–113.

Vassiliadis, P., Simitsis, A., and Skiadopoulos, S. (2002). Conceptual modeling for ETL processes. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 14–21. ACM.

Villata, S. and Gandon, F. (2012). Licenses compatibility and composition in the web of data. In *Proceedings of the Third International Conference on Consuming Linked Data-Volume 905*, pages 124–135. CEUR-WS. org.

Villazón-Terrazas, B. M. (2012). *A method for reusing and re-engineering non-ontological resources for building ontologies*, volume 12. IOS Press.

Vitvar, T., Kopeckỳ, J., Viskova, J., and Fensel, D. (2008). Wsmo-lite annotations for web services. *The semantic web: Research and applications*, pages 674–689.

Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G. (2009). Silk-A Link Discovery Framework for the Web of Data. *LDOW*, 538.

W3C OWL Working Group (2012). OWL 2 Web Ontology Language Document Overview (Second Edition). W3C recommendation, W3C. http://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

Waibel, M., Beetz, M., Civera, J., d'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J., Perzylo, A., et al. (2011). A world wide web for robots. *IEEE Robotics & Automation Magazine*, 18(2):69–82.

Wang, X., DeMartini, T., Wragg, B., Paramasivam, M., and Barlas, C. (2005). The MPEG-21 rights expression language and rights data dictionary. *IEEE Transactions on Multimedia*, 7(3):408–417.

Weber, I., Hoffmann, J., and Mendling, J. (2008). Semantic business process validation. In *Proceedings of the 3rd International Workshop on Semantic Business Process Management (SBPM'08), CEUR-WS Proceedings*, volume 472.

Weibel, S. (1999). The state of the Dublin Core metadata initiative: April 1999. *Bulletin of the Association for Information Science and Technology*, 25(5):18–22.

Weidlich, M., Decker, G., Großkopf, A., and Weske, M. (2008). BPEL to BPMN: The myth of a straight-forward mapping. *On the Move to Meaningful Internet Systems: OTM 2008*, pages 265–282.

Wikipedia (2014). Watson_(computer) — Wikipedia, the free encyclopedia. [Online; accessed 10-June-2014].

Wikipedia (2016a). Application Programmable Interface (API). [Online; accessed 24-Nov-2016].

Wikipedia (2016b). History of the world wide web. [Online; accessed 24-Nov-2016].

Wikipedia (2016c). World wide web. [Online; accessed 24-Nov-2016].

Wille, R. (1982). Restructuring lattice theory: an approach based on hierarchies of concepts. In *Ordered sets*, pages 445–470. Springer.

Wille, R. (2005). Formal concept analysis as mathematical theory of concepts and concept hierarchies. In *Formal concept analysis*, pages 1–33. Springer.

Wilson, M., Duce, D., and Simpson, D. (1989). Life cycles in Software and Knowledge Engineering: A comparative review. *The Knowledge Engineering Review*, 4(03):189–204.

Wintgens, L. J. (2005). Making Sense of Coherence. The Level Theory of Coherence. In *JURIX*, pages 23–24.

Wohed, P., van der Aalst, W. M., Dumas, M., ter Hofstede, A. H., and Russell, N. (2006). On the suitability of BPMN for business process modelling. In *International conference on business process management*, pages 161–176. Springer.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., et al. (2013). The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic acids research*, page gkt328.

Wroe, C., Goble, C., Goderis, A., Lord, P., Miles, S., Papay, J., Alper, P., and Moreau, L. (2007). Recycling workflows and services through discovery and reuse. *Concurrency and Computation: Practice and Experience*, 19(2):181–194.

Yuan, E. and Tong, J. (2005). Attributed based access control (ABAC) for web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE.

Zablith, F., d'Aquin, M., Brown, S., and Green-Hughes, L. (2011). Consuming linked data within a large educational organization. In *Proceedings of the Second International Conference on Consuming Linked Data-Volume 782*, pages 85–96. CEUR-WS. org.

Zablith, F., Fernandez, M., and Rowe, M. (2012). The OU linked open data: production and consumption. In *The Semantic Web: ESWC 2011 Workshops*, pages 35–49. Springer.

Zednik, S., Tilmes, C., and Hua, H. (2013). PROV-xml: The PROV xml schema. W3C note, W3C. http://www.w3.org/TR/2013/NOTE-prov-xml-20130430/.

Zhang, J., Tan, W., Alexander, J., Foster, I., and Madduri, R. (2011). Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 48–55. IEEE.

Zhang, Q., McCullough, J., Ma, J., Schear, N., Vrable, M., Vahdat, A., Snoeren, A. C., Voelker, G. M., and Savage, S. (2010). Neon: system support for derived data management. In *SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'10) No6*, volume 45(7). ACM.

Zinn, D. and Ludäscher, B. (2010). Abstract provenance graphs: anticipating and exploiting schema-level data provenance. *Provenance and Annotation of Data and Processes*, pages 206–215.

Zygiaris, S. (2013). Smart city reference model: Assisting planners to conceptualize the building of smart city innovation ecosystems. *Journal of the Knowledge Economy*, 4(2):217–231.